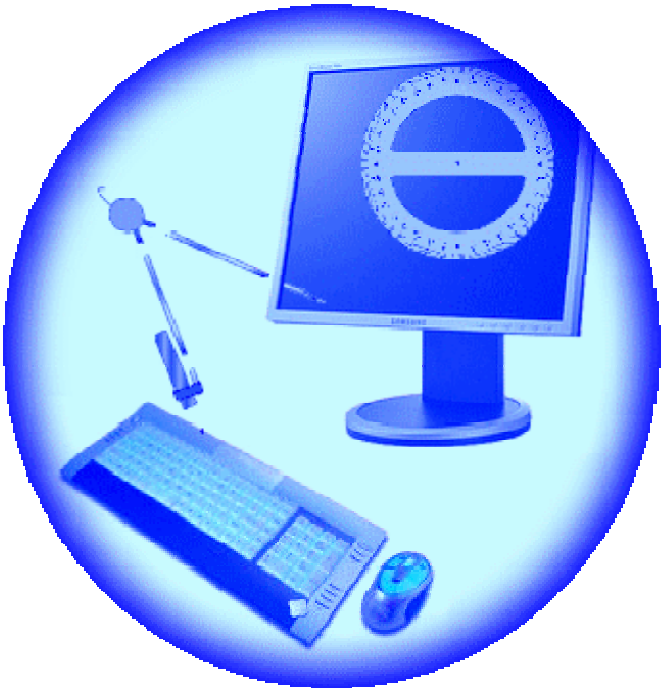
MATHÉMATIQUES APPLIQUÉES À L'INFORMATIQUE.



ENSEIGNEMENT SECONDAIRE SUPÉRIEUR DE TRANSITION.

Code : 012101U21D2

Année scolaire 2012/2013

CAPACITÉS PRÉALABLES REQUISES.

1. Capacités.

L'étudiant sera capable :

en informatique,
face à un système informatique connu, en respectant le temps alloué, les règles d'utilisation du système informatique et en utilisant les commandes appropriées :

- de mettre en en route le système informatique ;
- d'utiliser ses périphériques ;
- de mettre en œuvre des fonctionnalités de base du système d'exploitation en vue de la gestion de répertoires et de fichiers ;
- de créer et d'imprimer un fichier ;
- de clôturer une session de travail ;

en mathématiques,

- d'appliquer les règles et conventions du calcul algébriques ;
- de résoudre une équation du premier degré à une inconnue (type simple à coefficient numérique) ;
- de transformer une formule en fonction du résultat recherché ;
- d'utiliser le système métrique (prises de mesures et conversions).

2. Titre pouvant en tenir lieu.

Attestation de réussite de l'unité de formation « Introduction à l'informatique » de l'enseignement secondaire supérieur de transition et un certificat de l'enseignement secondaire inférieur ou un certificat de l'enseignement secondaire du deuxième degré.

PROGRAMME.

L'étudiant sera capable :

face à des situations concrètes,
d'acquérir et de mobiliser, d'une manière générale, les connaissances, les techniques et les méthodes pour :

- s'approprier le sens du vocabulaire mathématique et l'utiliser, d'une manière rigoureuse, dans diverses situations de la vie professionnelle ;

- découvrir l'implémentation des opérations arithmétiques élémentaires dans un système informatique ;
- mettre en œuvre une démarche de résolution de problèmes en utilisant les notions suivantes :
  - ♦ expression d'un nombre décimal en notation scientifique et vice-versa,
  - ♦ conversion entre les systèmes de numération décimale, binaire et hexadécimale,
  - ♦ opérations de l'arithmétique binaire (addition, soustraction, multiplication, division, complément à 2),
  - ♦ représentation des données numériques et non numériques dans un système informatique,
  - ♦ algèbre de Boole (proposition, conjonction, disjonction, négation, tables de vérité, loi de De Morgan, raisonnement et implication logique),
  - ♦ unités informatiques de mesure de quantité d'informations (bit, octet et multiples), de débit (bps, bauds), de fréquence (Hz et multiples),
  - ♦ utilisation et conversion des unités du système informatique,
  - ♦ représentation graphique d'une fonction à une variable,
  - ♦ représentation d'informations contenues dans un texte sous forme de tableau, schéma ou graphique,
  - ♦ interprétation des informations contenues dans un graphique en langage mathématique et en langage courant.

CAPACITÉS TERMINALES.

Pour atteindre le seuil de réussite, l'étudiant sera capable :

- d'exprimer un nombre décimal en notation scientifique et vice-versa ;
- de passer d'un système de numération à l'autre ;
- d'effectuer une addition dans les systèmes binaire et hexadécimal ;
- de résoudre un problème à l'aide de l'algèbre de Boole ;
- de représenter sous forme de tableau, schéma et graphique des informations contenues dans un texte.

Pour la détermination du degré de maîtrise, il sera tenu compte des critères suivants :

- le niveau de précision,
- la correction et la rigueur du vocabulaire utilisé,
- la pertinence des choix opérés.

1) NOTATION SCIENTIFIQUE.

a) INTRODUCTION.

Lorsqu'on utilise un tableur ou une calculatrice scientifique, on rencontre fréquemment des nombres exprimés sous la forme : 1.23E+6 ou -0.987E-17. Il s'agit d'un format d'expression des très petits et très grands nombres sous une forme compacte appelée *notation scientifique*. Elle est basée sur les puissances de 10. Le principe en est très simple.
Le nombre 1 000 est égal à 10 x 10 x 10, autrement dit on prend 3 fois le nombre 10 pour le multiplier par lui-même, on l'a élevé à la puissance 3 et l'on peut écrire que 1 000 est égal à 10^3 qui s'énonce « dix puissance trois ». Le nombre 3 s'appelle ici *l'exposant*. En continuant comme cela on aura :

$100 = 10^2$ soit 10 pris deux fois pour être multiplié par lui-même,

$10 = 10^1$ cas particulier à retenir,

$1 = 10^0$ autre cas particulier intéressant,

$0{,}1 = 10^{-1}$ l'exposant devient négatif,

$0{,}01 = 10^{-2}$ et ça continue …

b) MODE DE RÉSOLUTION.

Pour écrire le nombre 655,957 en notation scientifique, il faut le transformer en un produit de deux facteurs :

- Un facteur qui sera un nombre décimal dont :
  - ♦ la partie entière sera comprise entre 1 et 10,
  - ♦ la partie décimale contiendra tous les autres chiffres écrits dans le même ordre.
- L'autre facteur sera une puissance de 10.

	Partie entière	,	Partie décimale	X	Puissance de 10
Écriture décimale	**655**	**,**	**957**		
Notation scientifique	**6**	**,**	**55957**	**X**	**10^2**

c) EXEMPLES.

Écrire les nombres 2 378 et 0,051 en notation scientifique :

2 378	=	2,378	x	10^3
0,051	=	5,1	x	10^{-2}

d) ENTRAÎNEMENT.

1. Entourer les notations scientifiques parmi les écritures suivantes[1] :

$2{,}25 \times 10^4$ $\quad 0{,}14 \times 10^3$ $\quad 25{,}3 \times 10^4$ $\quad 2{,}5 \times 10^{-2}$ $\quad 10{,}34 \times 10^5$

$3^5 \times 10^4$ $\quad 5{,}3 \times 2^{10}$ $\quad 1{,}01 \times 10^{-4}$ $\quad 4{,}1 \times 2^3$ $\quad 10^2 \times 4{,}8$

2. Compléter pour que l'écriture du nombre soit sa notation scientifique[2] :

3 745,6 = 3,7……. x ……..

834,75 = 8, ……. x ……..

0,000004 = …….. x $10^{\cdots\cdots}$

2 715 000 = …….. x 10^6

0,0875 = 8,75 x ……..

3. Écrire les nombres en utilisant la notation scientifique[3] :

5 000 = ……..

1 000 000 = ……..

235,7 = ……..

125 000 = ……..

0,25 = ……..

0,00025 = ……..

1010 = ……..

2) NUMÉRATION ET BASES DE NUMÉRATIONS.

a) SYSTÈME DE NUMÉRATION.

La numération est :

- une méthode pour former les nombres
- une convention pour les écrire et les nommer.

Pour compter, nous dénombrons une à une les unités. À partir d'une certaine quantité d'unités on crée un ensemble d'une valeur déterminée auquel on donne un nom et que l'on met sur le côté pour compter les unités suivantes jusqu'à ce qu'on puisse les regrouper dans une autre ensemble de même taille. Les regroupements d'unités sont à leur tour regroupés en nouveaux ensembles qui portent un autre nom encore.

Exemple :

100 Cents = 1 €
1 000 gr = 1 kg, 1 000 kg = 1 T
60 sec = 1 min, 60 min = 1 h, 24 h = 1 jour
1' = 60'', 1 degré = 60', 1 tour = 360°
1 Pouce = 2,54 cm, 1 Pied = 12 Pouces, 1 Yard = 3 Pieds, 1 Mile = 1 760 Yards

Dans la vie courante, on essaie de compter par dizaines, centaines, milliers… nous essayons de n'utiliser qu'une seule base: la base 10.
Les chiffres arabes sont des signes particuliers pour désigner les neufs premiers chiffres et le zéro. Dix signes nous suffisent pour écrire tous les nombres. Les unités sont autant que possible regroupées par dizaines, les dizaines par centaines etc.

b) LES BASES DE NUMÉRATION.

Une base de numération est un nombre dont on utilise les puissances successives pour former d'autres nombres plus importants.
Ainsi, en base 10, les puissances successives sont Un ($1=10^0$), Dix ($10 = 10^1$), Cent ($100 = 10^2$), Mille ($1\,000 = 10^3$), Dix mille ($10\,000 = 10^4$) etc.
Le système décimal est le plus commun. Le choix de cette base n'est certainement pas indépendant du fait que nous ayons 10 doigts pour compter. Probablement que nous compterions en base 8 si nous étions des schtroumpfs

Il existe donc différentes bases de comptage :

Base 60	Système sexagésimal utilisé en Mésopotamie. Il nous en reste 60 minutes, 60 secondes.
Base 20	Le système vigésimal aurait été utilisé par nos ancêtres gaulois, il nous reste le « quatre-vingts ». Quatre-vingt-dix, soixante-quinze se basent sur des multiples de 20.
Base 12	Système duodécimal pour compter les mois, les heures et les œufs par douzaines.
Base 10	Celle que nous utilisons tous les jours.
Base 2	Incontournable en informatique. Sans elle, ce cours n'aurait pas lieu. Elle vient du fait que les ordinateurs sont construits à partir de composants qui, comme les contacts électriques, n'ont que deux états possibles : ouvert ou fermé, bloquant ou passant, 0 ou 1.
Base 16	Ressemble fort au binaire = notation plus concise.
Base 8	Très en vogue aux débuts de la micro-informatique.

3) ÉCRITURE DES NOMBRES ENTIERS OU NUMÉRATION DE POSITION.

a) ÉCRITURE DES NOMBRES.

On a vu que les nombres peuvent être formés en utilisant plusieurs bases de numération : la base 10 (décimale), la base 2 (binaire), etc. Le choix de la base ne suffit pas à expliquer comment nous écrivons les nombres. On utilisera pour cela une notation appelée numération de position. Pour comprendre de quoi il s'agit nous allons commencer par montrer ce que serait l'écriture des nombres sans cette numération de position.

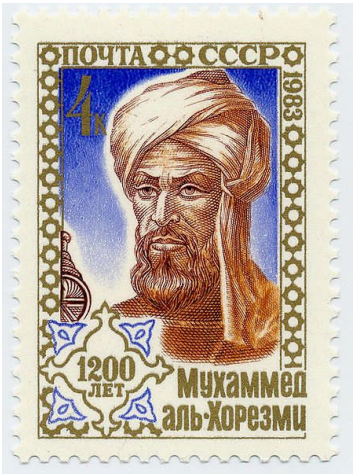
b) LES CHIFFRES ROMAINS.

Considérons le nombre 1975. On sait que les romains employaient eux-aussi le système décimal, la base 10, mais ils écrivaient leurs nombres différemment.

Voici comment 1975 s'écrit en chiffres romains : **MCMLXXV**

Cette écriture, plus compliquée mais encore utilisée dans certaines circonstances, se prête mal aux calculs écrits. Essayez donc de faire par écrit **MMX** moins **MCMLXXV** !

Pour les romains, mille, cent, dix et un ne pouvaient que s'écrire avec des signes différents car, sans le principe de la numération de position, ils ne pouvaient imaginer attribuer à leur chiffres des valeurs qui fluctuent selon leur position dans le nombre.



Ajoutez à cela le fait qu'ils ne connaissaient pas non plus le chiffre zéro. Ils n'avaient vraiment pas la chance que nous avons maintenant d'être familiarisés depuis notre plus tendre enfance à ces notions qui étonnamment n'ont été connues en occident qu'à partir du XII^{e} siècle alors que le mathématicien arabe Al-Khwarizmi[i], utilisait déjà le chiffre zéro au VIII^{e} siècle et qu'il était connue en Inde et probablement en Chine bien avant encore.

La numération de position combinée à l'utilisation du chiffre zéro nous permet de représenter les nombre de manière bien plus efficace et facilite grandement les opérations arithmétiques.

[i] **Al-Khawarizmi**, né vers 783, originaire de Khiva dans la région du Khwarezm, Ouzbékistan actuel [2] qui lui a donné son nom, mort vers 850 à Bagdad, est un mathématicien, géographe, astrologue et astronome perse, membre des Maisons de la sagesse dont les écrits, rédigés en langue arabe, ont permis l'introduction de l'algèbre en Europe.

c) NUMÉRATION DE POSITION.

Revenons au nombre 1975 écrit en base 10 comme nous en avons l'habitude.
La valeur que l'on attribue à chaque chiffre dépend du chiffre en lui-même et de sa position :

le 5 vaut 5 x 1,
le 7 représente des dizaines, il vaut 7 x 10,
le 9 qui suit représente des centaines, il vaut 9 x 100,
le 1 vaut 1 x 1 000.

Nous formons donc les nombres à l'aide d'une notation où la position est très importante.
Le chiffre le plus à droite représente des unités, celui directement à gauche, les dizaines, etc.
La position que le chiffre occupe dans le nombre est donc à considérer à partir de la droite.
Nous numéroterons donc ces positions en allant de droite à gauche. Ainsi le chiffre de droite aura toujours le même numéro quelle que soit la taille du nombre.
Cette numérotation commencera par le numéro 0 pour le premier chiffre (à droite donc) :

3	2	1	0
1	**9**	**7**	**5**

La règle qui permet de déterminer le poids d'un chiffre est la suivante :

Poids d'un chiffre = base^{position}

La valeur d'un chiffre est donc le produit de sa valeur propre et de son poids.
Le poids d'un chiffre est une puissance de la base et l'exposant y est la position du chiffre comptée de droite à gauche en commençant par les unités.
On peut aussi considérer que la position d'un chiffre dans un nombre entier est le nombre de chiffres qu'il y a à sa droite
Ainsi en base 10 :

En position 0 se trouvent les unités. Leur poids est **10^0**
En position 1 se trouvent les dizaines dont le poids est **10^1**
En position 2 : les centaines dont le poids est **10^2**
En position 3 : les milliers dont le poids est **10^3**
...
En position n : les milliers dont le poids est **10^n**

Voici ce que cela donne pour le nombre 1975 en décimal (Base 10) :

Le poids du chiffre 5 est $\mathbf{10^0}$, sa valeur est $5 \times 10^0 = 5 \times 1 = 5$
Le poids du chiffre 7 est $\mathbf{10^1}$, sa valeur est $7 \times 10^1 = 7 \times 10 = 70$
Le poids du chiffre 9 est $\mathbf{10^2}$, sa valeur est $9 \times 10^2 = 9 \times 100 = 900$
Le poids du chiffre 1 est $\mathbf{10^3}$, sa valeur est $1 \times 10^3 = 1 \times 1000 = 1000$

Positions	3	2	1	0
Chiffres décimaux	**1**	**9**	**7**	**5**
Valeurs de chaque chiffre	1×10^3	9×10^2	7×10^1	5×10^0
	1 000	900	70	5

$$1975 = 1 \times 10^3 + 9 \times 10^2 + 7 \times 10^1 + 5 \times 10^0$$

D'une manière plus théorique, on peut dire que la valeur d'un nombre **N** représenté par ***n*** chiffres en base **B** est la valeur numérique d'un polynôme du n-1^{ième} degré où **B** est la base et dont les coefficients sont entiers et inférieurs à **B** :

$$N = c_{n-1}B^{n-1} + \cdots + c_iB^i + \cdots + c_2B^2 + c_1B + c_0 = \sum_{i=0}^{i=n-1} c_iB^i$$

Ici en base 10, B = 10 et les coefficients c_{n-1}, c_{n-2}, … c_i, …, c_0 ont tous une valeur inférieure à 10.
La suite de ces coefficients $c_{n-1}\ c_{n-2}$ … $c_1\ c_0$ n'est autre que la suite des chiffres qui forment le nombre.

Mais revenons aux règles essentielles qu'implique ce qui a été vu jusqu'ici :

- Désormais, nous utilisons des numérations de position quelle que soit la base.
- Le chiffre le plus à droite représente toujours les unités.
- Les positions des chiffres se comptent de droite à gauche.
- La position du premier chiffre (celui de droite) est notée zéro.
- Ajouter un zéro à droite d'un nombre revient à multiplier ce nombre par sa base.

Petit quiz :

1) Dans quelle base de numération travaillent les ordinateurs ?[4]
   A. Binaire
   B. Hexadécimal
   C. Octal
   D. Décimal

2) Combien y a-t-il de chiffres différents en base 8 ?[5]
   A. 2
   B. 7
   C. 8
   D. 10

3) Le poids des chiffres en position 2 pour un nombre décimal est[6]
   A. 1
   B. 10
   C. 100
   D. 1 000

4) Le chiffre de position 1 dans le nombre 7865 est[7] :
   A. Le premier chiffre à gauche. Dans l'exemple c'est le chiffre 7.
   B. Le second chiffre à gauche. Le chiffre 8 dans cet exemple.
   C. L'avant dernier chiffre à droite. C'est le chiffre 6 dans l'exemple donné.
   D. Le dernier chiffre à droite. Dans l'exemple, c'est le chiffre 5.

5) Le chiffre en position 2 dans un nombre octal a pour poids[8] :
   A. 8
   B. 16
   C. 32
   D. 64

4) CALCUL DE LA VALEUR D'UN NOMBRE QUELLE QUE SOIT SA BASE.

La valeur d'un nombre est la somme des valeurs de chaque chiffre multiplié par leur poids respectif.
Cette règle reste toujours la même quelque soit la base considérée. Elle reste vraie tant que l'on applique les conventions de la numération de position.

Exemples :

1975_{10}
$= 1 \times 10^3 + 9 \times 10^2 + 7 \times 10^1 + 5 \times 10^0$
$= 1000 + 9000 + 70 + 5$
$= 1975$ Cela ne vous pose aucun problème en base 10 !
Alors, faites de même avec les autres bases.

1011011_2
$= 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$
$= 64 + 0 + 16 + 8 + 0 + 2 + 1$
$= 91$

175_8
$= 1 \times 8^2 + 7 \times 8^1 + 5 \times 8^0$
$= 1 \times 64 + 7 \times 8 + 5 \times 1$
$= 125$

72_{16}
$= 7 \times 16^1 + 2 \times 16^0$
$= 7 \times 16 + 2 \times 1$
$= 112 + 2$
$= 114$

5) NUMÉRATION BINAIRE.

En binaire la base est 2. Nous n'utilisons que deux chiffres 0 et 1.
Remarquez qu'en base 2, le chiffre 2 n'existe pas ; tout comme le chiffre 10 n'existe pas en base 10.
Il s'agit toujours d'une numération de position.
De droite à gauche nous avons donc les unités et ce que nous pourrions appeler les « deuzaines », les « quatraines », les « huitaines », les « seizaines », les « trente-deuzaines » etc.

a) POIDS DES BITS.

Prenons l'exemple du nombre binaire 10110.

Nous avons vu que le poids d'un chiffre dépend de sa position et de la base

$$\text{Poids} = \text{base}^{\text{position}}$$

Dans le cas du binaire, cela devient :
Le poids d'un bit dépend de sa position et de la base

$$\text{Poids} = 2^{\text{position}}$$

Appliquons cette règle à notre exemple $10110_{(2)}$

Positions	4	3	2	1	0
Chiffres binaires	**1**	**0**	**1**	**1**	**0**
Valeurs de chaque chiffre	1×2^4	0×2^3	1×2^2	1×2^1	0×2^0
	16	0	4	2	0

On a donc ici une « seizaine », une « quatraine » et une « deuzaine » soit 16 + 4 + 2 = 22.

b) UN PEU DE VOCABULAIRE…

Les codes binaires sont incontournables en informatique car l'information la plus élémentaire y est le **bit**.
Ce mot « bit » est formé par la fusion des mots ***Binary digit***
Ce qui en français se traduit par : **Chiffre binaire**

Un mot de 8 bits est appelé **Octet** (en français) et *Byte* (en anglais).

Les mots de 8, 16, 32 ou 64 bits sont courants. Écrits en binaire, ils sont plus lisibles si on laisse un espace entre les groupes de quatre bits comme ceci : 0100 0001
Un groupe de 4 bits est parfois appelé **Quartet** ou *nibble* mais ces termes sont peu utilisés.

Remarquez que l'on parle aussi de **Mots** binaires *(Word* en anglais) lorsque l'on veut spécifier la taille de ce mot. On dira de la sorte un mot de 32 bits ou un mot de 64 bits.

c) ZÉRO À GAUCHE :

Il est parfois intéressant de représenter les zéros non significatifs pour montrer la taille des codes transcrits. Il arrive que ces 0, à gauche des nombres, ne soient pas « non significatifs ». En effet, les codes binaires ne représentent pas toujours des valeurs numériques. Ce sont parfois simplement des codes qui ne représentent ni des quantités ni des valeurs ordinales. Inutile donc de faire de l'arithmétique avec ces codes. Dans ce cas cela n'a aucun sens non plus de vouloir les convertir en décimal et ce serait une erreur d'omettre l'écriture de ces zéros à gauche du code.

Les amateurs de formulations mathématiques se plairont à souligner que les bits nécessaires pour écrire la valeur N proviennent de la série des coefficients du polynôme suivant :

$$N = b_{n-1} \times 2^{n-1} + \cdots + b_i \times 2^i + \cdots + b_2 \times 2^2 + b_1 \times 2^1 + b_0 = \sum_{i=0}^{i=n-1} b_i 2^i$$

Les coefficients b_{n-1} … b_1 … b_2, b_1 et b_0 valent chacun **0** ou **1**.

d) PETIT QUIZ :

1) Quelle est en décimal la valeur du code binaire 0000 $1000_{(2)}$?[9]
2) Quelle est en décimal la valeur du code binaire 0000 $0101_{(2)}$?[10]
3) Quelle est en décimal la valeur du code binaire 0000 $0100_{(2)}$?[11]
4) Quelle est en décimal la valeur du code binaire 0001 $0000_{(2)}$?[12]
5) Le chiffre le plus à droite dans un nombre entier représente toujours les unités quelle que soit la base : décimale, binaire, octale, hexadécimale.
   Vrai ou faux ?[13]
   A. Vrai
   B. Faux
6) Convertir en décimal la valeur du code binaire 0000 $1010_{(2)}$?[14]
7) Donnez en décimal la valeur du nombre binaire 0000 $0010_{(2)}$?[15]
8) Convertir en décimal la valeur du code binaire 0000 $0111_{(2)}$?[16]

6) PESÉE D'UN NOMBRE EN BINAIRE.

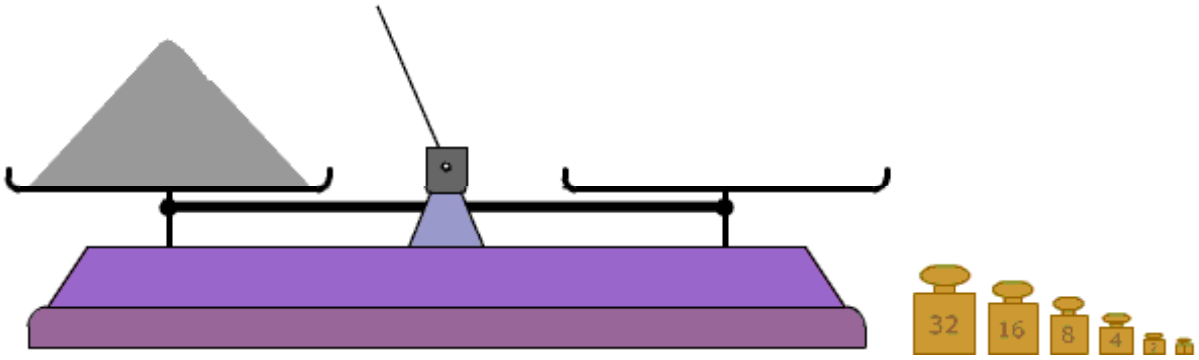
Chaque chiffre dans un nombre y a une importance, un « poids », qui dépend de sa position.

Il n'existe que deux chiffres en binaire : 0 et 1.
Le chiffre 0 a toujours la valeur 0 quelle que soit sa position.
La valeur du chiffre 1 est une puissance de 2, la position du bit comptée de droite à gauche est l'exposant de cette puissance.

Exercez-vous pour vous familiariser et progressivement apprendre quels sont les poids des bits.

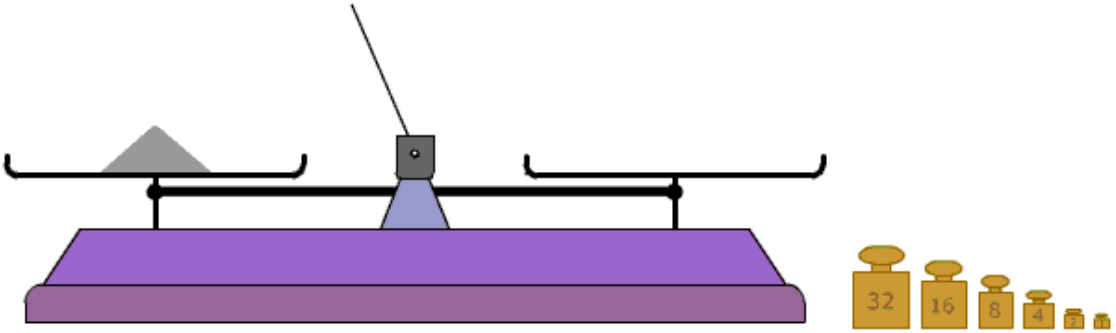
Quels poids faut-il mettre sur la balance pour l'équilibrer et obtenir cette valeur ?[17]

57



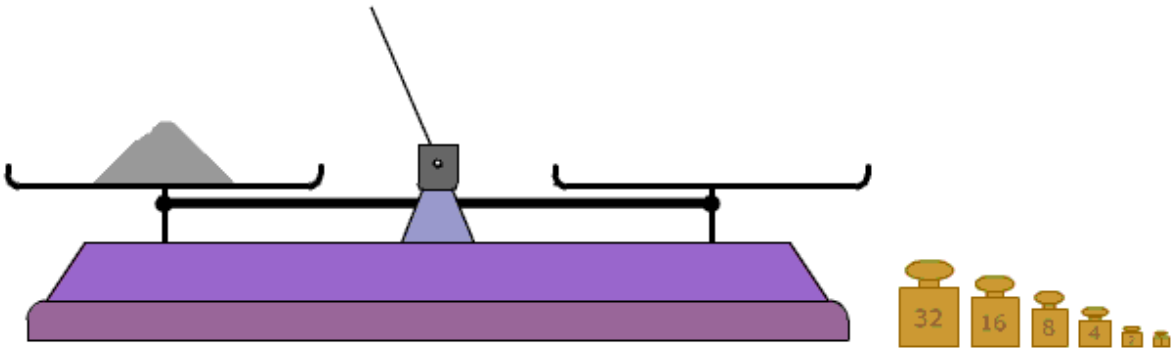
Quels poids faut-il mettre sur la balance pour l'équilibrer et obtenir cette valeur ?[18]

10



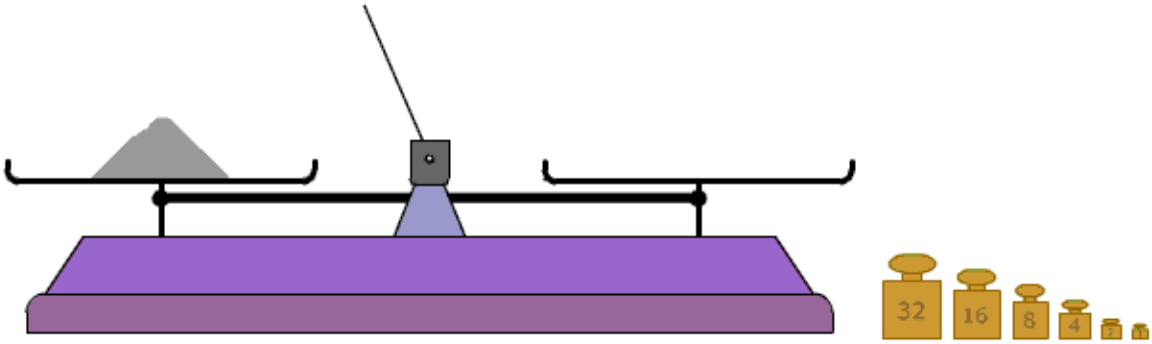
Quels poids faut-il mettre sur la balance pour l'équilibrer et obtenir cette valeur ?[19]

23



Quels poids faut-il mettre sur la balance pour l'équilibrer et obtenir cette valeur ?[20]

27



7) NUMÉRATION HEXADÉCIMALE.

a) CHIFFRES HEXADÉCIMAUX.

Il faut 16 chiffes pour écrire les nombres en base 16. Aux 10 chiffres du système décimal (0 à 9) ajoutons les 6 caractères A, B, C, D, E et F pour représenter ce que nous considèrerons ici comme étant les « chiffres » de 10 à 15.

Remarquez qu'en base 16, le chiffre 16 n'existe pas ; tout comme le chiffre 10 n'existe pas en décimal ni le chiffre 2 en binaire. Les principes de la numération de position sont applicables à toutes les bases et en particulier pour celle qui nous occupe ici, la base 16 :

- Le poids d'un chiffre dépend de sa position et de la base
- Poids = base^{position} ici en hexadécimal le poids = 16 ^{position}

Exemple : 1A2F hexadécimal ↦ Poids de chaque chiffre = $16^{position}$

Positions	3	2	1	0
Chiffres	**1**	**A**	**2**	**F**
Poids	$16^3 = 4\,096$	$16^2 = 256$	$16^{1=16}$	$16^0 = 1$
Valeur de chaque chiffre	1 x 4 096	10 x 256	2 x 16	15 x 1
	4 096	2 560	32	15
Valeur totale (comptée en décimal)	4 096 + 2 560 + 32 + 15 = 6 703			

On peut concevoir les nombres en base 16 comme une suite de coefficients d'un polynôme dont chaque terme est fait à partir des puissances successives de 16.

$$N = b_{n-1} \times 16^{n-1} + \cdots + b_i \times 16^i + \cdots + b_2 \times 16^2 + b_1 \times 16^1 + b_0 = \sum_{i=0}^{i=n-1} b_i 16^i$$

b) POURQUOI UTILISER LA BASE 16 ?

Les codes hexadécimaux sont bien pratiques en informatique. Ils représentent les codes binaires de manière compacte et nous évitent de devoir lire de longues enfilades de 0 et de 1 qui conviennent mieux aux ordinateurs qu'aux humains.

Un groupe de quatre bits permet de former 16 combinaisons différentes. On peut faire correspondre un chiffre hexadécimal à chacune de ces combinaisons de 4 bits. L'hexadécimal est en quelque sorte du binaire condensé.

Le code hexadécimal 1A2F est bien plus lisible que 0001 1010 0010 1111 en binaire

- Hexadécimal / Binaire – Mode d'emploi.

  Il est essentiel est de savoir compter jusqu'à 15 en binaire et en hexadécimal.
  Exercez-vous à reproduire le tableau ci-dessous jusqu'à ce que vous sachiez compter sans aucune difficulté de 0 à 15 en binaire et en hexadécimal.
  L'étape suivante de votre apprentissage sera de vous exercer à faire la correspondance entre les 16 codes binaires et les codes correspondants en hexadécimal.

Décimal	Binaire	Hexa
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

- Conversion Hexadécimal ↦ Binaire.

  Remplacer chaque chiffre hexadécimal par le code de 4 bits correspondant.

  Exemples :

  $7A6C_{(16)} = 0111\ 1010\ 0110\ 1100_{(2)}$
  $1234_{(16)} = 0001\ 0010\ 0011\ 0100_{(2)}$

- Conversion binaire ↦ Hexadécimal.

  Grouper les bits par tranche de 4 en commençant à partir de la droite, puis chercher dans la table ci-dessus quel chiffre hexadécimal correspond à chaque quartet.

  Exemple :

  $10100011011110_{(2)} = 0010\ 1000\ 1101\ 1110_{(2)} = 28DE_{(16)}$

c) EXERCICES.

1) Qu'est-ce qu'un bit ?[21]
   A. Une impulsion électrique.
   B. Une unité informatique.
   C. Un chiffre binaire.
   D. 8 bytes.

2) Dans quelle base les nombres sont-ils traités au niveau du processeur ?[22]
   A. Binaire.
   B. Octal.
   C. Décimal.
   D. Hexadécimal.

3) Combien de codes différents peut-on écrire avec 4 bits ?[23]
   ……

4) Quelle est la plus grande valeur qu'on puisse écrire avec 8 bits ?[24]
   ……

5) Que vaut 2 exposant 10 ?[25]
   ……

6) Pourquoi utilise-t-on la base hexadécimale en informatique ?[26]
   A. Parce que cela prend moins de place en mémoire.
   B. Car les informaticiens et tous les êtres humains en général ont plus facile de manipuler l'hexadécimal que le binaire.
   C. Car le processeur travaille en hexadécimal.
   D. Car c'est en hexadécimal que les données sont stockées en mémoire.

7) Comment écrit-on seize en hexadécimal ?[27]
   A. 1
   B. F
   C. 10
   D. 16
   E. 1F

8) Comment le code binaire 0000 1111 s'écrit-il en hexadécimal ?[28]
   ……

9) Comment appelle-t-on un code de 8 bits ?[29]
   A. Un quartet
   B. Un quatuor
   C. Un octet
   D. Un byte
   E. Un double mot

10) Que vaut le code hexadécimal A0 ?[30]
    Donnez votre réponse en base 10, bien entendu !
    ……

8) NOMBRES DE CODES POSSIBLES AVEC N CHIFFRES EN BASE QUELCONQUE.

a) EN DÉCIMAL (= EN BASE 10).

1 chiffre ➪ 10 codes différents (de 0 à 9)
2 chiffres ➪ 100 codes différents (de 00 à 99)
3 chiffres ➪ 1000 codes différents (de 000 à 999)
...
n chiffres ➪ 10^n codes différents

b) EN HEXADÉCIMAL (= EN BASE 16).

1 chiffre ➪ 16 codes différents (de 0 à F_{16})
2 chiffres ➪ 16^2 = 256 codes différents (de 00 à FF)
3 chiffres ➪ 16^3 = 4096 codes différents (de 000 à FFF)
...
n chiffres ➪ 16^n codes différents

c) EN BINAIRE (= EN BASE 2).

1 chiffre ➪ 2 codes différents (0 et 1)
2 chiffres ➪ 2^2 = 4 codes différents (de 00 à 11_2)
3 chiffres ➪ 2^3 = 8 codes différents (de 000 à111_2)
...
n bits ➪ 2^n codes différents

d) TAILLES DES NOMBRES ENTIERS ET NOMBRE DE CODES RÉALISABLES.

Mot de 1 byte = 8 bits appelé *byte* ou *octet*
➪ 2^8 = 256 codes possibles
Mot de 2 bytes = 16 bits parfois appelé *word* , *short* ou *integer*
➪ 2^{16} = 65536 codes possibles
Mot de 4 bytes = 32 bits souvent appelé *long*
➪ 2^{32} = 4 milliards de codes possibles
Mot de 8 bytes = 64 bits ➪ 2^{64} = 16 10^{18} = 16 milliards de milliards de codes possibles

e) CONCLUSION.

Le nombre de codes possibles avec N chiffres en base B est B^N

9) PUISSANCES DE 2.

a) PRÉFIXES POUR REPRÉSENTER LES PUISSANCES DE 2^{10} OU 10^3.

Nous sommes amenés en informatique à devoir chiffrer des grandeurs très grandes et d'autres très petites. La pratique du système métrique nous a habitués à exprimer ces nombres à l'aide de multiples de 10 et même souvent de 1000. Cela correspond à notre habitude de regrouper les chiffres par trois comme dans 1 000 ou 1 000 000 = 10^3 et 10^6

Pour les grands nombres, les puissances successives de 10^3 portent ces noms :

Kilo 1 k = 10^3
Méga 1 M = 10^6
Giga 1 G = 10^9
Tera 1 T = 10^{12}
Peta 1 P = 10^{15}
Exa 1 E = 10^{18}

Les petits nombres s'expriment au moyen de puissances de 10^{-3} :

milli 1 m = 10^{-3}
micro 1 µ = 10^{-6}
nano 1 n = 10^{-9}
pico 1 p = 10^{-12}
femto 1 f = 10^{-15}

Exercices.

1) Combien y a-t-il de µs dans une ms ?[31]
2) Les physiciens utilisent l'Angström comme unité pour mesurer les très petites dimensions.
   C'est le cas par exemple pour les dimensions des atomes. Un Angström est un dix-milliardième de mètre. Comment peut-on écrire cette distance en ne se référant qu'aux unités vues ci-dessus ?[32]

b) POUR LES INFORMATICIENS, 1 K EST-CE 1 000 OU 1 024 ?

Nous savons que $2^{10} = 1024$.
Ce nombre proche de 1000 est souvent désigné par le préfixe "kilo".

- Quand il s'agit de dimension de mémoires, on parle de KB (kilo bytes) ou de Ko (kilo octet) pour dénombrer des multiples de 1024 bytes. De même 1 MB ou 1 Mo = 1024x1024 bytes quand on parle de tailles de mémoire car le nombre de cellules mémoire dans n composant est toujours une puissance de 2 et donc un multiple de 2^{10} ou 2^{20}.
- Dans les autres cas, quand les kilos, les mégas et autres gigas ne concernent pas la mémoire tous ces préfixes représentent des multiples de 1000. 1 kHZ = 1000 Hz. 20 Go sur un disque = 20 milliards d'octets et non pas 20×1024×1024×1024.

Notez que le préfixe kilo s'écrit toujours avec un k minuscule quand sa valeur est 1000.
Exemple : 1 kHz = 1 000 Hertz ici **1k = 1 000**

Les informaticiens écrivent souvent ce préfixe avec une lettre majuscule pour la valeur 1 024.
Exemple : 1Ko = 1 024 octets ici **1K = 1 024**

c) CALCULS APPROXIMATIFS DE 2^N AVEC N > 10.

$2^{10} \approx 10^3$ car 1 024 ≈ 1 000

Si on accepte cette approximation, il est alors possible de calculer mentalement ce que fait 2^n même si n > 10

Exemple que vaut 2^{24} ?
$2^{24} = 2^4 \times 2^{10} \times 2^{10} = 16 \times 1\,024 \times 1\,024 = 16\text{ M} \approx 16\,000\,000$

Conclusions : Puisque $2^{10} \approx 10^3$
on a directement $2^{20} \approx 10^6$, $2^{30} \approx 10^9$, $2^{40} \approx 10^{12}$, etc.

Exercices.

1) Calculer 2^{12}, 2^{32}, 2^{16}, 2^{27} et 2^{36}.[33]

2) On a depuis peu utilisé tous les codes de numéros d'immatriculations composés de 3 lettres suivies de 3 chiffres pour les plaques belges. Combien de nouveaux codes d'immatriculation pourra-t-on faire en plaçant cette fois d'abord 3 chiffres puis 3 lettres ?[34]

Quiz.

1) Que vaut 2^{20} ?[35]
   A. 2 K
   B. 20 K
   C. 1 M
   D. 2 M
   E. 12 K

2) Que vaut 2^{24} ?[36]
   A. 16 K
   B. 64 K
   C. 1 M
   D. 16 M
   E. 12 K

3) Combien de couleurs peut-on coder avec un mot de 32 bits ?[37]
   A. 256
   B. 4 millions
   C. 16 millions
   D. 4 milliards

4) Que vaut approximativement 2^{12} ?[38]
   A. 1 K
   B. 2 K
   C. 4 K
   D. 8 K
   E. 12 K

5) Une transmission à 100 Mb/s laisse passer chaque seconde :[39]
   A. 100 bits / seconde
   B. 100 000 bits / seconde
   C. 100 000 000 bits / seconde
   D. 100 000 000 000 bits / seconde

6) Que vaut 2^{10} ?[40]

……

7) Que vaut 2^{16} ?[41]

A. 16 K
B. 64 K
C. 1 M
D. 16 M
E. 12 K

8) Parmi les réponses qui suivent, laquelle vaut approximativement 2^{10} ?[42]

A. 10^2
B. 10^3
C. 10^6
D. 10^9

10) CONVERTIR UN NOMBRE ENTIER DANS UNE BASE QUELCONQUE.

a) RAPPEL : VALEUR DE CHAQUE CHIFFRE.

Nous avons vu précédemment comment convertir un nombre de base quelconque en base 10. Il suffit pour ce faire d'avoir compris le principe de la numération de position. Chaque chiffre a une valeur qui dépend du chiffre lui-même et de sa position. On obtient la valeur d'un nombre en additionnant les valeurs des chiffres qui le composent.

Une autre manière d'exprimer la même chose est de dire qu'en lisant un nombre de droite à gauche on rencontre les puissances successives de la base :

- unités, « deuzaines », « quatraines », huitaines, seizaines etc. pour le binaire (base 2)
- unités, huitaines, « soixante-quatraines » etc. pour l'octal (base 8)
- unités, dizaines, centaines, milliers, etc. pour le décimal (base 10)
- unités, seizaines, « deux-cent-cinquante-sixaines », etc. pour l'hexadécimal (base 16)

Voyons à présent comment coder dans une base B quelconque un nombre N dont on connaît la valeur décimale, c'est à dire son écriture en base 10. Il faut pour cela dénombrer les puissances successives de la base :

- le nombre d'unités, de « deuzaines », de « quatraines » etc. pour convertir en binaire
- le nombre d'unités, de seizaines, etc. pour convertir en base 16

Il y a pour ce faire deux méthodes : de « gauche à droite » et de « droite à gauche ».

b) MÉTHODE INTUITIVE : DE GAUCHE À DROITE.

- Quelle est la plus grande puissance p de la base B que l'on puisse retrouver dans N et combien de fois y retrouve-t-on la valeur de B^p ?
  Cela donne le premier chiffre à gauche, en position p.

  Exemple : Soit à convertir $420_{(10)}$ en base 16.

  $420_{(10)}$ est supérieur à 16^2,
  $16^2 = 256$ va une fois dans 420 ⇒ le chiffre le plus à gauche est 1

  Reste à représenter $420 – 1 \times 16^2 = 420 - 256 = 164$ unités

- On répète la même question tant que le reste est supérieur à la base.
  $164_{(10)}$ est supérieur à 16^1,
  16^1 va 10 fois dans 164 ⇒ chiffre suivant est $A_{(16)}$

Reste 164 – 10 x 16 = 4 unités

- Le nombre d'unité qui reste inférieur à B est le chiffre le plus à droite autrement dit en position 0 si les positions sont numérotées de droite à gauche.

c) MÉTHODE SYSTÉMATIQUE : DE DROITE À GAUCHE.

Commençons par rechercher la valeur du premier chiffre à droite. Ce chiffre, les unités, est le reste de la division du nombre N à convertir par la base qu'ici nous appelons B. Ce chiffre en position 0 a un poids égal à la base exposant zéro = $B^0 = 1$ = l'unité.

En divisant à nouveau le quotient de la division précédente par la base on obtient le chiffre de position 1 dont le poids est B^1 = la base.

Des divisions répétées par la base donnent successivement les chiffres de poids B^0, B^1, B^2, B^3, B^4 etc. ce qui nous permet d'écrire le nombre de droite à gauche.

Exemples :

1° Convertir **$1830_{(10)}$** en binaire ⇒ divisions successives par 2

1830	: 2	=	915	reste	0	0	x	2^0	=	0 unité
915	: 2	=	457	reste	1	1	x	2^1		
457	: 2	=	228	reste	1	1	x	2^2		
228	: 2	=	114	reste	0	0	x	2^3		
114	: 2	=	57	reste	0	0	x	2^4		
57	: 2	=	28	reste	1	1	x	2^5		
28	: 2	=	14	reste	0	0	x	2^6		
14	: 2	=	7	reste	0	0	x	2^7		
7	: 2	=	3	reste	1	1	x	2^8		
3	: 2	=	1	reste	1	1	x	2^9		
1	: 2	=	0	reste	1	1	x	2^{10}		

0 C'est fini, il ne reste plus rien à diviser.

Le résultat est : $111\ 0010\ 0110_{(2)}$

2° Convertir **$1830_{(10)}$** en hexadécimal ⇒ divisions successives par 16

1830	: 16	=	114	reste	6	6	x	16^0	=	6 unités
114	: 16	=	7	reste	2	2	x	16^1	=	2 seizaines
7	: 16	=	0	reste	7	7	x	16^2		
0	C'est fini, il ne reste plus rien à diviser.									

Le résultat est $726_{(16)}$, ce qui concorde bien avec la valeur trouvée en binaire.

d) CONCLUSIONS.

Ce procédé fonctionne pour toutes les bases mais en informatique seuls nous concernent le binaire et l'hexadécimal, parfois mais plus rarement l'octal (base 8).
La conversion en binaire est la plus facile, le reste vaut 0 pour les nombres pairs et 1 pour les nombres impairs.
On a donc avantage à convertir d'abord en binaire. Le passage en hexadécimal comme nous l'avons vu au début du cours n'est plus alors qu'un jeu d'enfant.

e) AUTRE MÉTHODE POUR CONVERTIR UNE BASE B EN BASE 10 = « MÉTHODE DE HORNER ».

Nous avons vu comment calculer la valeur d'un nombre quelle que soit la base utilisée pour le représenter. Nous additionnions les valeurs obtenues en calculant les valeurs de chaque chiffre compte tenu de leurs positions dans le nombre.
La méthode qui suit donne le même résultat.
Montrons comment cela marche pour le binaire mais la méthode est valable quelle que soit la base.

Voici l'algorithme :

```
Lire la valeur du chiffre à gauche
Répéter tant qu'il reste des chiffres à droite
{
   Multiplier par la base
   Ajouter le chiffre suivant
}
```

Exemples :

1° $1101_{(2)} = \Big(\big(\big((\mathbf{1} \times 2) + \mathbf{1} \big) \times 2 \big) + \mathbf{0} \Big) \times 2 \Big) + \mathbf{1} = 13$

2° $123_{(8)} = \Big(\big((\mathbf{1} \times 8) + \mathbf{2} \big) \times 8 \Big) + \mathbf{3} = 83$

3° $20C_{(16)} = \Big(\big((\mathbf{2} \times 16) + \mathbf{0} \big) \times 16 \Big) + \mathbf{12} = 524$

f) EXERCICES.

1° Rechercher par la méthode de Horner :[43]

$203_{(8)}$ =
$101010_{(2)}$ =
$20A_{(16)}$ =

2° Méthode au choix[44] :

$166_{(10)}$ = ……(16)
$100_{(10)}$ = ……(2)
$100_{(10)}$ = ……(16)
$1023_{(10)}$ = ……(16)
$1023_{(10)}$ = ……(2)
$C0CA_{(16)}$ = ……(10)
$1011011_{(2)}$ = ……(10)
$236_{(8)}$ = ……(10)
$FFF_{(16)}$ = ……(10)

11) NOMBRES SIGNÉS.

Nous avons jusqu'à présent parlé de **nombres entiers naturels**. Ils ne peuvent par nature qu'être positifs ou nuls. Envisageons maintenant les **nombres entiers relatifs** ou autrement dit, munis d'un signe « + » ou « - »

En décimal,
+1, +2, +3 etc. sont des nombres positifs. Ils sont supérieurs à 0 (n >0)
-1, -2, -3 etc. sont des nombres négatifs. Ils sont inférieurs à 0 (n < 0)

De même en binaire,
+1, +10, +11, +100, +101 etc. sont des nombres binaires positifs,
-1, -10, -11, -100, -101 etc. sont des nombres binaires négatifs.

Le problème est que les circuits électroniques digitaux ne peuvent enregistrer que des 0 ou des 1 mais pas de signes + ou -. Le seul moyen est alors de convenir que si un nombre est susceptible d'être négatif on lui réserver un bit pour indiquer le signe. Reste à déterminer le bit qui dans un nombre binaire conviendrait le mieux pour symboliser le signe et quelle valeur de ce bit (0 ou 1) conviendrait le mieux pour représenter le signe "plus" ou le signe "moins".

Observons d'abord le fait que les nombres codés en machine ont une dimension fixe :

- Sur papier, les nombres ont des dimensions variables :
  - L'addition de deux nombres de 2 chiffres donne un nombre de 2 ou 3 chiffres.
  - La multiplication de deux nombres de 2 chiffres donne des nombres de 3 ou 4 chiffres.
- En machine par contre, les nombres ne sont pas extensibles. Ils ont des dimensions fixes. C'est exactement ce que nous avons avec certain compteurs.

Dans une voiture par exemple, le compteur kilométrique s'il ne possède que 6 chiffres ne pourra indiquer plus de 999 999 km.
De même, dans les ordinateurs les nombres (binaires) ont aussi des dimensions fixes de 1, 2, 4 ou 8 octets.

Revenons à l'exemple de la voiture et imaginez un compteur kilométrique qui compte les km en marche avant et qui les décompte en marche arrière. Que pourrait-on lire sur un compteur d'une voiture neuve (compteur initialement à 000 000) si elle parcourt 1 km en marche arrière ? Le compteur décompte 1 km et affiche donc ... 999 999 km ! Ce code correspond parfaitement à la valeur –1 puisqu'on obtient 0 si on lui ajoute à nouveau 1.

$x + 1 = 0 \Rightarrow x = -1 \Rightarrow$ dans ce cas ci 999 999 équivaut à -1

On exploite cette caractéristique étrange qui est due au fait que ce nombre à une dimension finie (6 chiffres décimaux)

De même, quel serait le code d'un nombre de 8 bits pour représenter la valeur –1 ?
Le code $1111\ 1111_{(2)} = FF_{(16)}$ convient puisque, si on ajoute 1 à ce nombre, on obtient $0000\ 0000_{(2)} = 00_{(16)}$, le bit de report déborde à gauche, il sort de l'espace qui est réservé au nombre et est donc ignoré.

Le bit le plus à gauche du mot binaire est celui qui va représenter le signe. Signe négatif si ce bit vaut 1, signe positif quand ce bit vaut 0.

Si on admet que le nombre peut représenter des valeurs négatives, on parle de nombres « signés ».

Comme pour les nombres « non signés », on peut représenter $2^8 = 256$ codes avec 8 bits mais ici le bit de gauche est le signe

1 = signe moins
0 = signe plus

Il y a donc moyen de représenter :

- 128 codes avec le bit de signe à 1
  ce sont 128 nombres négatifs
  (de –1 à – 128)
- 128 codes avec le bit de signe à 0
  le nombre 0 et 127 nombres positifs
  (de 1 à +127)

Nombre de 8 bits

Lu en hexadécimal	Lu en binaire	Lu en décimal signé	Lu en décimal non signé
7F	0111 1111	+127	127
7E	0111 1110	+126	126
…	…	…	…
10	0001 0000	+16	16
0F	0000 1111	+15	15
0E	0000 1110	+14	14
0D	0000 1101	+13	13
0C	0000 1100	+12	12
0B	0000 1011	+11	11
0A	0000 1010	+10	10
09	0000 1001	+9	9
08	0000 1000	+8	8
…	…	…	…
02	0000 0010	+2	2
01	0000 0001	+1	1
00	0000 0000	+0	0
FF	1111 1111	-1	255
FE	1111 1110	-2	254
FD	1111 1101	-3	253
FC	1111 1100	-4	252
FB	1111 1011	-5	251
FA	1111 1010	-6	250
F9	1111 1001	-7	249

…	…	…	…
86	1000 1001	-122	134
85	1000 0101	-123	133
84	1000 0100	-124	132
83	1000 0011	-125	131
82	1000 0010	-126	130
81	1000 0001	-127	129
80	1000 000	-128	128

a) COMMENT CALCULER LES CODES DES NOMBRES NÉGATIFS ?

Le calcul se fait en deux étapes :

1° Calcul du complément à 1 = Remplacer tous les 0 par des 1 et tous les 1 par des 0.
2° Calcul du complément à 2 = Ajouter 1 au complément à 1

Exemple :

Comment écrire – 4 en binaire ou en hexadécimal ?
+ 4 = $0000\ 0100_{(2)}$
Le complément à 1 de ce code est $1111\ 1011_{(2)}$
Ajoutons 1 à ce code pour obtenir son complément à 2
$1111\ 1011_{(2)} + 1 = 1111\ 1100_{(2)} = FC_{(16)} = -\ 4_{(\text{décimal signé})}$

Cas particuliers :

Le complément à 2 de 0 est encore 0
Le complément à 2 de 80H est aussi 80H ! Les nombres négatifs et positifs ne sont pas répartis symétriquement.

Avec un byte la valeur minimum est –128 contre +127 pour la valeur positive.

N.B. : Le complément à 1 est aussi appelé « *complément logique »*" ou « *complément restreint ».*
De même, certains désignent le complément à 2 par l'expression « *complément arithmétique »*.

b) ANALOGIE EN DÉCIMAL.

25	Existe-t-il un complément arithmétique de 17
-17	tel que 25 + (Complément de 17) = 25 - 17 ?
	Oui, à condition de décréter que comme dans une machine les nombres ont
08	une taille fixe au-delà de laquelle les reports sont ignorés.

Puisque deux chiffres suffisent pour écrire 25, 17 et 08 nous limitons la taille de ces nombres à 2 caractères.

La question devient : Quel nombre faut-il ajouter à 25 pour que la réponse se terminer par les chiffres 08 ?

Ce nombre est 83. En effet **25 + 83 =** 1 **08** mais on ignore le 1 à gauche puisque nous avons décidé de donner une taille fixe de deux chiffres pour les nombres de cet exemple.

83 est donc dans ce cas le complément arithmétique de 17.

Comment trouver ce complément arithmétique en base 10 ?

La méthode ressemble fort au calcul du complément à 1 comme en binaire suivi de l'addition d'une unité. Ici, en décimal, le complément restreint sera un complément à 9.

Complément à 9 : 99 -17 = 82
Complément arithmétique : 82 +1 = 83
99 – 17 + 1 = 100 – 17 = 83

c) LA VALEUR DU BIT DE SIGNE.

Le bit de signe est le bit le plus significatif du code (MSB *Most Significant Bit*), celui qui est le plus à gauche. Dans le cas d'un nombre de n bits numérotés de 0 à n-1, c'est le bit n-1. Bien souvent on se contente de constater que ce bit est à 1 pour en conclure que le nombre considéré est négatif. La valeur absolue de ce nombre est alors déterminée en calculant le complément arithmétique de son code.

Une autre manière d'envisager la chose serait de considérer que le bit n-1 a, contrairement aux autres bits, une valeur négative : -2^{n-1}

Exemple :

Si un byte est considéré comme un code signé le bit 7 quand il est à 1 vaut -128. Si le byte est considéré comme non signé, le poids du bit 7 est simplement 2^7 = 128.
Ainsi -123 = -128 + 5 = 80H + 5 = 85H

d) EXTENSION DE LA TAILLE D'UN NOMBRE SIGNÉ.

Pour étendre la taille d'un nombre non signé, on ajoute des 0 à sa gauche.
Pour étendre la taille d'un nombre signé, on ajoute sur la gauche des bits identiques au bit de signe.

Exemples :

-4 code sur un octet = $FC_{(16)}$ sur deux octets ce code devient $FFFC_{(16)}$
$FC_{(16)} = 1111\ 1100_{(2)} \equiv FFFC_{(16)} = 1111\ 1111\ 1111\ 1100_{(2)}$
de même
+4 en un octet = $04_{(16)}$ sur deux octets = $0004_{(16)}$
$04_{(16)} = 0000\ 0100_{(2)} \equiv 0004_{(16)}0000\ 0000\ 0000\ 0100_{(2)}$

e) EXERCICES.

➢ Quiz – Nombres signés.

1) Quel est dans un octet le bit qui sert de bit de signe ?[45]
   A. Bit 0
   B. Bit 1
   C. Bit 7
   D. Bit 8
   E. Bit 15

2) Quelle est la plus grande valeur positive que l'on puisse écrire un nombre signé de 8 bits ?[46]
   A. 99
   B. 127
   C. 128
   D. 255
   E. 256

3) Quelle est la valeur décimale du nombre signé $FF_{(16)}$? soit 1111 1111 en binaire ?[47]
   ……

4) Que vaut le nombre non signé $FF_{(16)}$? soit 1111 1111 en binaire ?[48]
   ……

5) Que vaut le nombre signé $80_{(16)}$ ou 1000 0000 en base 2 ?[49]
   ……

6) Que donne le calcul 0 moins 1 s'il est fait en hexadécimal avec des nombres de 1 octet ?[50]

……

7) Que donne le calcul 0 moins 1 s'il est fait en hexadécimal avec des nombres de 2 octets ?[51]

……

8) Le code de 8 bits 7F en hexadécimal a-t-il la même valeur si ce nombre est signé ou non ?[52]
   A. Oui
   B. Non

9) Quelle est la plus grande valeur positive que l'on puisse écrire avec un nombre signé de 2 octets ?[53]
   A. 32 767
   B. 32 768
   C. 65 535
   D. 65 536

12) OPÉRATIONS ARITHMÉTIQUES EN BINAIRE.

Nous nous limitons dans ce chapitre au cas des nombres entiers.

Aussi étonnant que cela puisse paraître, il faut bien l'avouer, ce n'est qu'à de très rares occasions qu'un informaticien est amené à faire par écrit des calculs en binaire ou en hexadécimal. On dispose bien souvent d'une machine, une calculatrice ou un ordinateur, pour réaliser de telles opérations. Il est cependant important de connaître les mécanismes de ces opérations pour saisir comment ces machines fonctionnent, tout comme nous nous sommes attachés à comprendre les méthodes des changements de bases ou du codage des nombres signés.

Les calculs en binaire (ou en hexadécimal) peuvent toujours se faire exactement de la même manière que ceux que nous faisions à l'école primaire en base 10.

a) ADDITION.

« Un plus un » fait deux, c'est un fait indépendant du mode de représentation des nombres.

En binaire, deux s'écrit 10 $\quad 1_{(2)} + 1_{(2)} = 10_{(2)} \quad \Rightarrow$ « 1 + 1 = 2, j'écris 0 et je reporte 1 »

De même (1 + 1 + 1 = 3) $\quad 1_{(2)} + 1_{(2)} + 1_{(2)} = 11_{(2)} \quad \Rightarrow$ « 1 + 1 + 1 = 3, j'écris 1 et je reporte 1 »

Pour le reste 0 + 0 = 0, 0 + 1 = 1 et 1 + 0 = 1
Ces trois derniers calculs n'engendrent aucun report.

Vous en savez assez maintenant pour additionner par écrit deux nombres de ***n*** bits.
Il suffit d'aligner convenablement ces deux nombres, l'un au-dessus de l'autre, une colonne à droite pour les unités puis successivement vers la gauche les colonnes des « deuzaines », des « quatraines » etc.
Additionnez ensuite les bits en commençant par la droite sans oublier de noter les reports.

Exemple :

```
  1 0 1 1 0 1 0 1          1     1 1 1
+     1 0 0 1 1 1    ↦   + 1 0 1 1 0 1 0 1
-----------------          + 1 0 0 1 1 1
                           -----------------
                           1 1 0 1 1 1 0 0
```

b) SOUSTRACTION.

Si vous y tenez, vous pouvez appliquer à nouveau la même méthode qu'à l'école primaire.
Aligner le nombre à soustraire sous le premier nombre puis on effectue la soustraction en commençant par les chiffres à droite. Si le chiffre du dessous est trop important, il faut enregistrer une « retenue » qu'on retranche dans le calcul de la colonne suivante.

Exemple :

```
      10
   3  5
-  1  7
   1
-------
   1  8
```

On peut appliquer le même procédé en binaire :

```
        10  10      10
   1  0  0  1  0  1
-     1  1  0  1  1
   1  1     1
-------------------
   0  0  1  0  1  0
```

Sachez toutefois que même les machines ne se donnent jamais ce mal.
Plutôt que de faire une soustraction, elles additionnent le complément du terme à soustraire.

Plutôt que de soustraire un nombre, nous allons ajouter son complément. La méthode n'a de sens que pour des nombres ayant une taille finie. Prenons l'exemple ci-dessus et fixons la taille des nombres à 8 bits.

		1	1	1		1		1	
Nous additionnerons donc le complément de	0001 1011		0	0	1	0	0	1	0	1
Complément à 1 ↦	1110 0100	+	1	1	1	0	0	1	0	1
Complément à 2 ↦	1110 0101	1	0	0	0	0	1	0	1	0

c) MULTIPLICATION.

Les multiplications écrites se font de la même manière en binaire qu'en décimal. Il suffit de connaître la table de multiplication par 0 et par 1.

X	0	1
0	0	0
1	0	1

Exemple :

			1	1	0	1	0	1	1	0
		x						1	0	1
1	1	1	1		1					
			1	1	0	1	0	1	1	0
		0	0	0	0	0	0	0	0	
	1	1	0	1	0	1	1	0		
1	0	0	0	0	1	0	1	1	1	0

d) DIVISION.

Ici aussi, nous pouvons utiliser la même méthode que lors des calculs écrits en décimal.
En binaire, l'écriture des multiples de 2 se termine par le chiffre 0. Pour diviser par 2, il suffit donc d'enlever le zéro à droite du nombre.

Exemple :

$\frac{10}{2} = 5 \rightarrow \frac{1010_{(2)}}{10_{(2)}} = 101_{(2)}$

1	0	0	0	0	1	0	1	1	1	0	1	0	1					
-	1	0	1	.	.	.	.	.	.	.	1	1	0	1	0	1	1	0
		1	1	0	.	.	.	.	.	.								
	-	1	0	1	.	.	.	.	.	.								
				1	1	0	.	.	.	.								
			-	1	0	1	.	.	.	.								
						1	1	1	.	.								
					-	1	0	1	.	.								
							1	0	1	.								
						-	1	0	1	.								
									0	0								

e) CONCLUSION.

Mis à part les additions, il est fort rare de faire des opérations arithmétiques en binaire « à la main » comme il est tout aussi rare de faire de l'informatique sans machine. Le but du chapitre était de comprendre comment se font les opérations afin de pouvoir imaginer ce qui se passe dans les machines, et d'être capable de comprendre et d'interpréter les résultats.

13) LOGIQUE BOOLÉENNE.

a) INTRODUCTION.

La logique, si l'on considère l'histoire des sciences, est une discipline très ancienne dont les traces les plus éloignées remontent aux philosophes Grecs. Logos en grec se traduit par parole et/ou raison. Aristote, parait-il, s'amusait à voir comment étaient construits les faux raisonnements ... Mais, ce n'est pas de cette logique là que nous traiterons ici.

La logique qui va nous occuper dans les pages suivantes concerne l'étude de fonctions logiques élémentaires qui traitent des variables binaires par des méthodes aussi systématiques que les mathématiques. Le but est ici de comprendre et démystifier autant que se peut la manière dont les ordinateurs effectuent les opérations logiques de base puis d'entrevoir comment ces fonctions peuvent être combinées pour obtenir des résultats de plus en plus complexes comme par exemple les opérations arithmétiques.

b) VARIABLES LOGIQUES.

Dans les systèmes digitaux (systèmes informatiques et autres automatismes numériques) toutes les données sont traitées et enregistrées à partir d'éléments d'informations binaires.

Ces informations binaires à la manière des contacts électriques n'ont que deux états possibles : un contact électrique est ouvert ou fermé, de même le bit est une information élémentaire qui ne peut prendre que deux valeurs 0 et 1.

Dans la suite de ce cours les variables logiques sont représentées par des lettres. Les valeurs de ces variables sont à priori indéterminées, elles sont « variables ». Ainsi, si l'on écrit l'équation logique « S = A » cela signifie que la variable logique S a exactement la même valeur que la variable A mais rien ne précise quelle est la valeur de A (0 ou 1 ?) On dira que A est une **variable binaire** puisqu'elle ne peut prendre à un instant donné qu'une des deux valeurs 0 ou 1.

NB. Un parallèle peut être fait avec les variables algébriques comme par exemple dans la fonction algébrique « y = 2x + 5 » Elle nous indique comment calculer y pour chaque valeur de x mais la valeur de x n'est pas figée. C'est ici une variable algébrique.

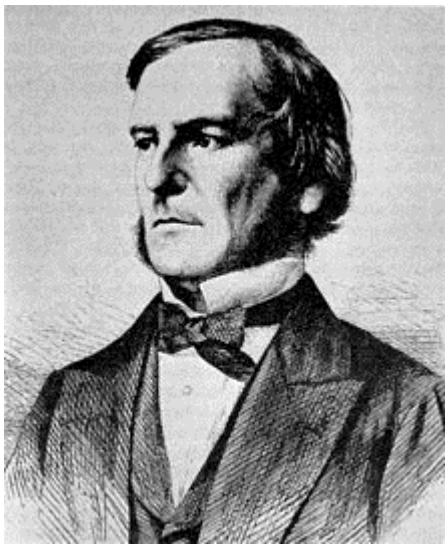
Les opérations logiques sont en informatique aussi courantes si pas plus que les opérations arithmétiques. La logique combinatoire tout comme l'arithmétique repose sur quelques opérations élémentaires.

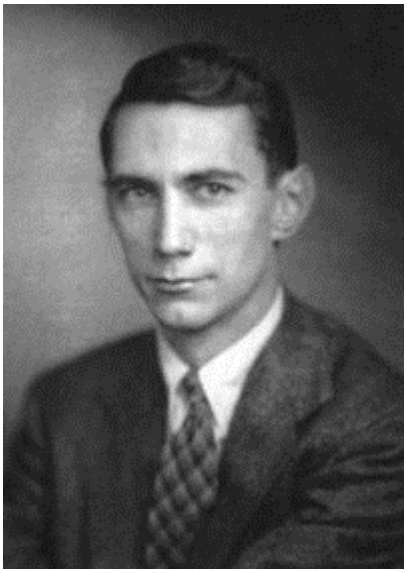
- En arithmétique, ces opérations sont l'addition, la soustraction, la multiplication et la division (**+, -, *, /**). Il est possible à partir de là

d'imaginer toutes les autres opérations telles que les exposants, les racines, les logarithmes etc.

- En logique, les opérations fondamentales sont le **ET**, le **OU** et le **NON**.

  Nous utiliserons des signes particuliers pour représenter ces trois opérations fondamentales lors d'écriture d'équations logiques. C'est **George Boole**[ii], un mathématicien britannique, qui le premier eu l'idée de reprendre des notations algébriques pour créer les bases de ce qui sera la logique informatique. Nous ferons donc de la **logique booléenne** et aussi de **l'algèbre booléenne** en écrivant des **équations logiques** pour exprimer les relations entre les variables logiques appelées aussi **variables booléennes**.





Cette logique a trouvé après George Boole ses premières applications dans les circuits électriques. C'est **Claude Shannon**[iii] , un autre père fondateur des théories à la base de l'informatique, qui entreprit de mettre en équation les circuits électriques où des relais électriques considérés comme des variables logiques en agissent sur des contacts ouverts (0) ou fermé (1).

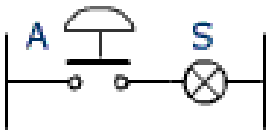
[ii] **George Boole**, né le 2 novembre 1815 à Lincoln (Royaume-Uni) et mort le 8 décembre 1864 à Ballintemple (Irlande), est un logicien, mathématicien et philosophe britannique. Il est le créateur de la logique moderne, fondée sur une structure algébrique et sémantique, que l'on appelle algèbre de Boole en son honneur.

[iii] Claude Shannon est né le 30 avril 1916 à Gaylord dans le Michigan dont il fréquente l'université. Et où il obtient une licence de mathématiques et de physique. Il entre au M.I.T et dans sa thèse de Master, « Une analyse symbolique des circuit à relais et de commutation » il utilise l'algèbre de Boole pour concevoir les circuits de commutation. Il apporte ainsi un outil théorique aux concepteurs de circuits logiques, qui servira aussi bien pour les circuits à relais que pour les circuits intégrés.

14) FONCTIONS LOGIQUES DE BASE.

La manière la plus simple de comprendre les fonctions logiques est de se les représenter par des schémas électriques qui comportent un ou plusieurs contacts et une lampe. Cette lampe s'allume « ***à condition*** » que les contacts électriques y laissent passer le courant. C'est dans l'expression de cette condition que va intervenir la logique.

Le schéma ci-contre traduit la condition la plus simple : la lampe s'allume si le bouton poussoir A est actionné.
Autrement dit (S = 1) si (A = 1)



Le fonctionnement de ce circuit s'exprime par l'équation logique : **S = A**.

Il n'y a que deux cas possibles. Ils sont représentés dans cette **table de vérité** :

A	**S**
0	0
1	1

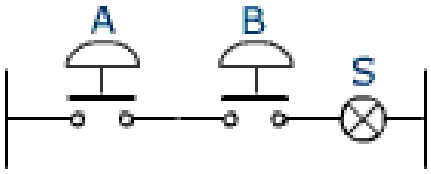
Une table de vérité a pour le rôle de montrer la correspondance entre la sortie et toutes les combinaisons de valeurs que peuvent prendre la ou les entrées.

Le contenu de la table de vérité s'obtient en imaginant toute les configurations possibles pour le contact A (0 = contact relâché, 1 = bouton pressé). À chacun des états du contact A on vérifie d'après le schéma électrique ou d'après l'équation logique si la lampe est éteinte (0) ou allumée (1).

Plaçons maintenant deux contacts dans le circuit. La condition nécessaire pour allumer la lampe dépend de la manière dont les contacts sont connectés. Suivant les cas, la condition pour allumer la lampe fait appel aux opérateurs logiques ET ou OU.

a) LA FONCTION ET.

La lampe s'allume si on active simultanément les contacts A et B.



(S = 1) si (A = 1) ET (B = 1).

- Table de vérité.

A	B	S
0	0	0
0	1	0
1	0	0
1	1	1

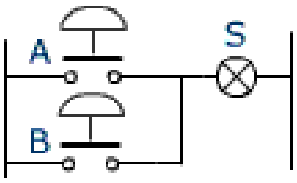
Le contenu de la table de vérité s'obtient en imaginant toute les configurations possibles pour le contact A (0 = contact relâché, 1 = bouton pressé). À chacun des états du contact A on vérifie d'après le schéma électrique ou d'après l'équation logique si la lampe est éteinte (0) ou allumée (1).

- Équation logique de la fonction ET.

S = A . B

L'opérateur ET est représenté dans l'équation logique par un point. Ce signe convient parfaitement puisque la fonction ET donne le même résultat qu'une multiplication.

b) LA FONCTION OU.



La lampe s'allume si on active le contact A ou le contact B.

(S = 1) si (A = 1) OU (B = 1)

- Table de vérité.

A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

- Équation logique de la fonction OU.

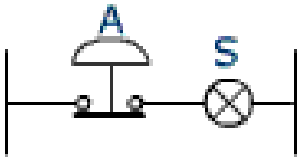
$$S = A \dot{+} B$$

L'opérateur OU est représenté dans l'équation logique par un signe plus surmonté d'un point. Observons les trois premières lignes de la table de vérité, le résultat de l'opération OU y est semblable au résultat d'une addition. Le résultat de 1 ou 1 diffère cependant de 1+1. Nous mettons un point au-dessus du signe « + » pour indiquer que l'opération n'est pas analogue à une addition.

c) LA FONCTION NON.

Les contacts que nous avons utilisés jusqu'ici, sont des contacts « normalement ouverts ». Quand le bouton poussoir est relâché (quand A = 0) le courant ne passe pas.

Nous utilisons maintenant un contact « normalement fermé » pour illustrer la fonction NON. Au repos, le courant passe mais il se coupe quand le contact est activé (quand A = 1).



- Table de vérité.

A	S
0	1
1	0

- Équation logique de la fonction NON.

$S = \overline{A}$ qui se lit : « S = Non A »

15) COMBINAISONS DE FONCTIONS LOGIQUES.

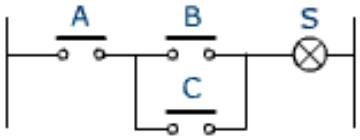
a) COMBINAISONS DES 3 FONCTIONS DE BASE.

Les trois fonctions de base AND, OR et NOT se combinent de multiples façons. Ces combinaisons peuvent facilement être illustrées par des montages de contacts électriques.

- Correspondance entre schémas et équations logiques.

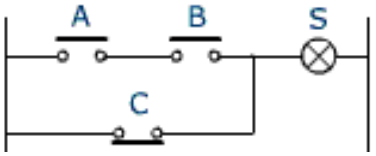
  À chaque schéma imaginable correspond une équation. La correspondance entre un schéma et une fonction logique est systématique :
  - Des montages en parallèle correspondent à la fonction OU
  - Des montages en série correspondent à la fonction ET
  - Un contact normalement fermé représente la fonction NON

Exemples :



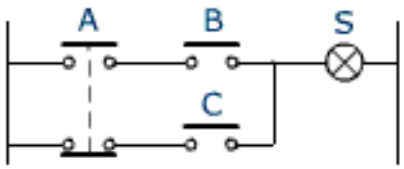
S = A . (B + C)

A	B	C	S
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



S = (A . B) + $\bar{C}$)

A	B	C	S
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1



$S = (A \cdot B) \dotplus (\overline{A} \cdot C)$

A	B	C	S
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

- Tables de vérité.

Pour tracer la table de vérité, que ce soit à partir d'un schéma ou d'une équation, il faut commencer par dénombrer les variables. Les exemples ci-dessus comportent chacun 3 variables (A, B et C).
Il faut remarquer qu'une même variable peut intervenir plusieurs fois dans une équation ou sur plusieurs contacts dans le schéma. C'est le cas de la variable A dans le troisième exemple.
On trace ensuite une table avec toutes les combinaisons possibles des *n* variables. Cela donne 2^n combinaisons. $8 = 2^3$ dans nos exemples. Le plus facile est de noter toutes ces combinaisons en considérant qu'elles forment des nombres écrits en binaire et de les aligner une à une comme si l'on comptait en base 2. On est alors certain en faisant de la sorte de n'avoir oublié aucun cas.

Reste maintenant de déduire les valeurs à indiquer dans la colonne de droite.
Elles résultent du fonctionnement du schéma de contacts ou du calcul de ce que donne l'équation logique.

- Déduction à partir du schéma de contacts :
  On considère qu'un contact est actionné si la variable correspondante vaut 1. Le contact est dans sa position de repos (comme représenté sur le schéma) lorsque la variable correspondante vaut 0. Attention à ne pas confondre le fait qu'un contact soit actionné et le fait qu'il soit fermé. Un contact normalement ouvert laisse passer le courant quand il est actionné mais un contact normalement fermé laisse passer le courant quand il n'est pas actionné.

- Calcul à partir de l'équation logique :
  On substitue les variables par les valeurs qui leurs sont attribuées aux différentes lignes de la table de vérité.

b) EXERCICES.

1° Quelle fonction logique représente cette table de vérité ?

A	B	S
0	0	0
0	1	0
1	0	0
1	1	1

A. AND
B. OR
C. XOR
D. NAND
E. NOR

2° Quelle fonction logique représente cette table de vérité ?

A	B	S
0	0	1
0	1	1
1	0	1
1	1	0

A. AND
B. OR
C. XOR
D. NAND
E. NOR

3° Quelle est la table de vérité qui correspond à cette équation logique ?

$$S = \overline{A \cdot B}$$

A.

A	B	S
0	0	1
0	1	1
1	0	1
1	1	0

B.

A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

C.

A	B	S
0	0	1
0	1	0
1	0	0
1	1	0

D.

A	B	S
0	0	1
0	1	1
1	0	1
1	1	0

4° Son équation logique est :

$$\mathbf{S = A \oplus B}$$

Voici sa table de vérité :

A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

Quelle est cette fonction logique ?

A. AND
B. OR
C. XOR
D. NAND
E. NOR

5° Quelle est la table de vérité (A, B, C ou D) qui correspond à cette équation logique ?

$$S = \overline{A \dot{+} B}$$

A.

A	B	S
0	0	1
0	1	1
1	0	1
1	1	0

B.

A	B	S
0	0	1
0	1	0
1	0	0
1	1	0

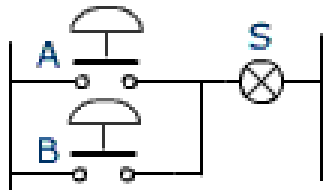
C.

A	B	S
0	0	0
0	1	0
1	0	0
1	1	1

D.

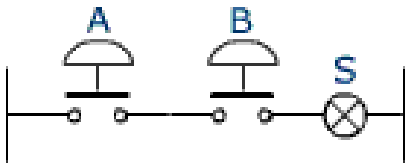
A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

6° Quelle est cette fonction ?



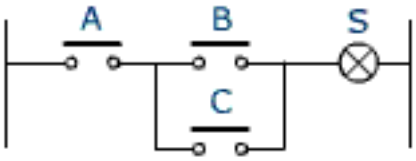
A. AND
B. OR
C. XOR
D. NAND
E. NOR

7° Quelle fonction logique est illustrée par ce schéma ?



A. AND
B. OR
C. XOR
D. NAND
E. NOR

8° Parmi les cinq équations proposées, quelle est celle qui correspond au schéma ci-dessus ?



A. **S = A + B** →S = A ou B)

B. **S = A . B** →S = A et B

C. **S = (A . B) + $\bar{C}$** →S = (A et B) ou pas C

D. **S = A . (B + C)** → S = A et (B ou C)

E. **S = (A . B) + ($\bar{A}$. C)** → S = (A et B) ou (Pas A et C)

16) LES PORTES LOGIQUES.

Nous avons jusqu'ici utilisé des boutons poussoirs et une lampe pour illustrer le fonctionnement des opérateurs logiques. En électronique digitale, les opérations logiques sont effectuées par des portes logiques. Ce sont des circuits qui combinent les signaux logiques présentés à leurs entrées sous forme de tensions. On aura par exemple 5V pour représenter l'état logique 1 et 0V pour représenter l'état 0.

a) FONCTIONS LOGIQUES DE BASE.

- Porte AND.

Symbole américain de la porte ET à deux entrées.

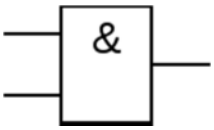
Équation logique de la fonction ET.

$S = A \cdot B$

Table de vérité d'une porte ET à deux entrées.

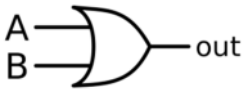
A	B	S
0	0	0
0	1	0
1	0	0
1	1	1

Symbole international de la porte ET à deux entrées.



- Porte OR.

Symbole américain de la porte OU à deux entrées.



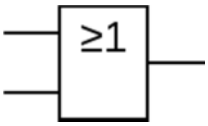
Équation logique de la fonction OU.

$S = A \dot{+} B$

Table de vérité d'une porte OU à deux entrées.

A	B	S
0	0	0
0	1	1
1	0	1
1	1	1

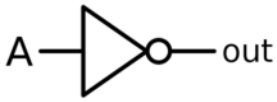
Symbole international de la porte OU à deux entrées.



- Porte NOT.

En électronique, une porte NON est plus communément appelée *inverseur*. Le cercle utilisé sur la représentation est appelé « bulle », et on l'utilise généralement dans les diagrammes pour montrer qu'une entrée ou une sortie est inversée.

Symbole américain d'une porte inverseuse.



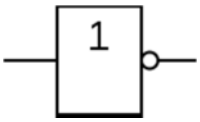
Équation logique de la fonction NOT.

$S = \overline{A}$

Table de vérité d'une porte inverseuse.

A	S
0	1
1	0

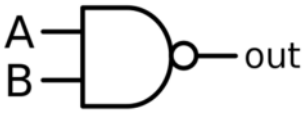
Symbole international d'une porte inverseuse.

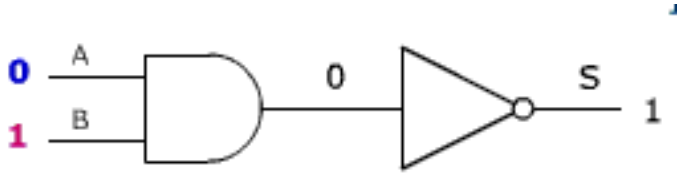


b) COMBINAISONS DES FONCTIONS LOGIQUES DE BASE.

- Porte NAND (NON ET).

Symbole américain.





Équation logique.

$S = \overline{A \cdot B}$

Symbole international.

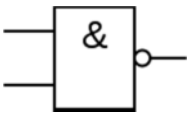
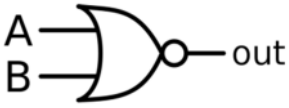


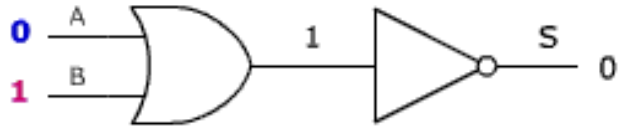
Table de vérité.

A	B	S
0	0	1
0	1	1
1	0	1
1	1	0

- Porte NOR (NON OU).

Symbole américain.





Équation logique.

$S = \overline{A \dot{+} B}$

Symbole international.

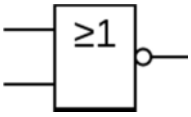
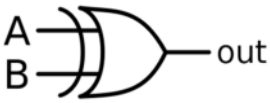


Table de vérité.

A	B	S
0	0	1
0	1	0
1	0	0
1	1	0

- Porte XOR (OU exclusif).

Symbole américain.



La sortie est à 1 si une seule des deux entrées vaut 1.

Équation logique.

$S = A \oplus B$

Symbole international.

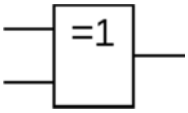
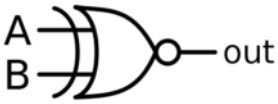


Table de vérité.

A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

- Porte XNOR (NON OU exclusif).

Symbole américain.



Équation logique.

$S = \overline{A \oplus B}$

Symbole international.

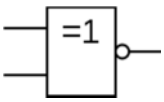


Table de vérité.

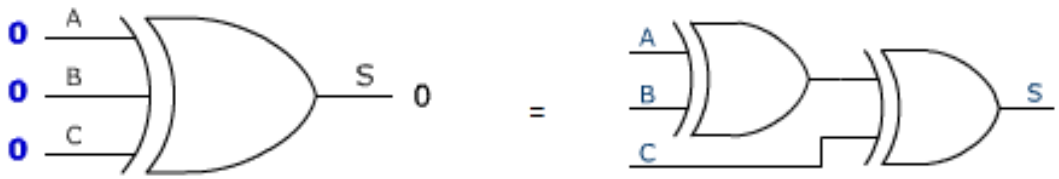
A	B	S
0	0	1
0	1	0
1	0	0
1	1	1

- Porte XOR à plusieurs entrées.

Pour calculer le résultat de **S = A xor B xor C**, il faut d'abord faire l'opération entre deux termes, puis refaire un OU exclusif entre le résultat obtenu et le troisième terme.

Ce qui se traduit par : **S = (A xor B) xor C** ou par **S = A xor (B xor C)**

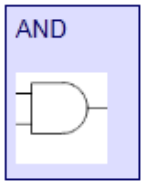
On constate que l'appellation « OU exclusif » n'est tout à fait exact que pour deux variables. Avec trois variables, le résultat vaut 1 si une d'entre elles ou toutes les trois valent 1.



Le résultat vaut 1 si le nombre d'entrées à 1 est impair.

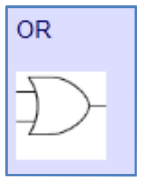
c) Exercice d'association[54].

Quels sont les tables de vérité de ces fonctions ?



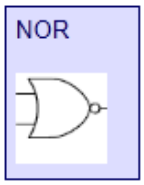
■ ■

A	B	S
0	0	0
0	1	1
1	0	1
1	1	0



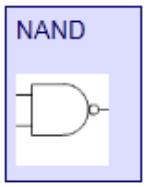
■ ■

A	B	S
0	0	0
0	1	0
1	0	0
1	1	1



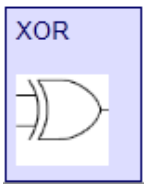
■ ■

A	B	S
0	0	0
0	1	1
1	0	1
1	1	1



■ ■

A	B	S
0	0	1
0	1	1
1	0	1
1	1	0



■ ■

A	B	S
0	0	1
0	1	0
1	0	0
1	1	0

17) UNITÉS INFORMATIQUES DE MESURE DE QUANTITÉ D'INFORMATIONS (BIT, OCTET ET MULTIPLES), DE DÉBIT (BPS, BAUDS), DE FRÉQUENCE (HZ ET MULTIPLES).[iv]

a) LE SYSTÈME DE NUMÉRATION BINAIRE

Il existe plusieurs unités de mesure en informatique. Il est utile de comprendre ces mesures à même titre qu'il est utile de connaître les grammes, les centimètres ou les kilomètres dans la vie de tous les jours. Le binaire est cependant l'alphabet des ordinateurs puisque ces derniers comptent en base deux. Cela signifie qu'ils n'utilisent que deux chiffres pour faire des nombres, à savoir 0 et 1. Les systèmes informatiques actuels étant construits à l'aide de circuits intégrés, ils ne peuvent fonctionner que selon une logique de deux états: le courant ne passe pas (0) ou passe (1) dans le transistor. Ainsi, les dix premiers nombres du système de numération binaire sont: 0, 1, 10, 11, 100, 101, 110, 111, 1000 et 1001. À l'opposé, l'homme utilise dix chiffres pour faire des nombres (0, 1,2,...,9).

L'unité de mesure la plus utilisée en informatique est sans aucun doute le bit (Binary digit). Cette unité élémentaire d'information permet de représenter les deux valeurs du système de numération binaire. La manipulation des bits dans un ordinateur s'effectuer généralement en les regroupant en blocs de 8 bits appelés octets (bytes en anglais). Considéré isolément, 1 bit a donc peu de signification, contrairement à 1 octet qui peut servir à représenter des chiffres et des lettres de l'alphabet par exemple. Alors que seulement 4 bits d'informations sont nécessaires pour représenter chaque nombre du système décimal, 7 bits d'informations sont nécessaires pour représenter chaque caractère de l'alphabet international défini dans les années soixante aux États-Unis, puis repris par les organismes de normalisation des transmissions internationales de données. Connu sous le nom de code ASCII (American Standard Code for Information Interchange), cet alphabet comprend 128 caractères distincts. En français, la norme ISO latin-1 reprend les 128 premiers caractères de l'ASCII et y ajoute des caractères accentués.

b) LES MULTIPLES INFORMATIQUES

Dans le domaine informatique, les multiplicateurs Kilo, Méga, Giga et Téra ne sont pas toujours reliés à une puissance de 10, car parfois ils sont reliés à une puissance de 2, notamment lorsqu'il est question de capacité ou de vitesse. Lorsque les professionnels de l'informatique ont remarqué que 2^{10} (2x2x2…x2 = 1024) équivalait sensiblement à 10^3 (10x10x10 = 1000), ils ont décidé d'utiliser le préfixe "kilo" pour désigner 1024. Par exemple 1 kilooctet équivaut donc à 1024 octets et non à 1000 octets. Ils ont continué d'utiliser cette "convention" au fil des ans…et ce même si les capacités de stockage des ordinateurs n'ont cessé d'augmenter. Le résultat aujourd'hui est que le grand public ignore ce qu'est réellement un Mégaoctet. Les

[iv] http://micro.info/chronique/chronique.php?Id=825

fabricants de mémoire vive utilisent le terme Mégaoctet pour désigner 2^{20} = 1 048 576 octets alors que les fabricants d'unité de disque utilisent le même terme pour désigner 10^6 = 1 000 000 octets, entretenant ainsi une certaine confusion. Pour remédier à ce problème, l'IEC (International Electrotechnical Commission) a récemment proposé quatre termes pour corriger l'utilisation abusive des multiplicateurs du Système International à savoir Kibi (2^{10}), Mébi (2^{20}), Gibi (2^{30}) et Tébi (2^{40}).

c) LES UNITÉS DE CAPACITÉ

Unité	Symbole	Équivalence	Exemple
Bit	bit	Unité de base	Le code ASCII à 7 bits
Octet	o	1 octet = 8 bits	Un caractère occupe 1 octet
Kilooctet	Ko	1 Ko = 2^{10} octets	Un fichier texte de 10 Ko
Mégaoctet	Mo	1 Mo = 2^{20} octets	128 Mo de mémoire vive
Gigaoctet	Go	1 Go = 2^{30} octets	Un disque dur de 40 Go
Téraoctet	To	1 To = 2^{40} octets	Un système RAID de 1 To

d) LES UNITÉS DE FRÉQUENCE

Unité	Symbole	Équivalence	Exemple
Hertz	Hz	Unité de base	Le taux de rafraîchissement d'un moniteur est de 75 Hz
Kilohertz	KHz	1 KHz = 10^3 Hz	Une pièce de musique échantillonnée à 44,1 KHz
Mégahertz	MHz	1 MHz = 10^6 Hz	Un processeur Pentium II cadencé à 300 MHz
Gigahertz	GHz	1 GHz = 10^9 Hz	Un processeur Pentium 4 cadencé à 2 GHz

e) LES UNITÉS DE TEMPS

Unité	Symbole	Équivalence	Exemple
Seconde	s	Unité de base	Formater une disquette 3½ prend environ 120 s
Milliseconde	ms	1 ms = 10^{-3} s	Un disque dur avec un temps d'accès de 12 ms
Microseconde	µs	1 µs = 10^{-6} s	L'exécution d'instructions sur un système de type CISC peut prendre 1µs
Nanoseconde	ns	1 ns = 10^{-9} s	Le temps d'accès de la mémoire vive est de 60 ns
Picoseconde	ps	1 ps = 10^{-12} s	Le temps d'accès d'une mémoire cache peut être aussi bas que 500 ps

f) LES UNITÉS DE VITESSE

Unité	Symbole	Équivalence	Exemple
Baud	Bd	Unité de base	Les premiers modems atteignaient 300 Bd
Bit/seconde	bps[v]	1 bps = 1 Bd	Un fax opère à 9600 bps
Kilobit/seconde	Kbps	1 Kbps = 10^3 Bd	Un modem téléphonique opère à 56 Kbps
Mégabit/seconde	Mbps	1 Mbps = 10^6 Bd	Une carte réseau Ethernet atteint 10 à 100 Mpbs
Gigabit/seconde	Gbps	1 Kbps = 10^9 Bd	Les routeurs du RISQ qui lient Québec et Montréal opèrent à 2,5 Gbps
Térabit/seconde	Tbps	1 Tbps = 10^{12} Bd	Le routeur le plus rapide a atteint le cap du 1 Tbps

[v] Il ne faut pas confondre bps (bit/seconde) et Bps (byte/seconde ou octet/seconde). Cette unité est utilisée en anglais pour désigner la vitesse de transfert de données. Ex.: un fichier est téléchargé à partir d'Internet à une vitesse de 100 KBps, soit 100 Ko/s.

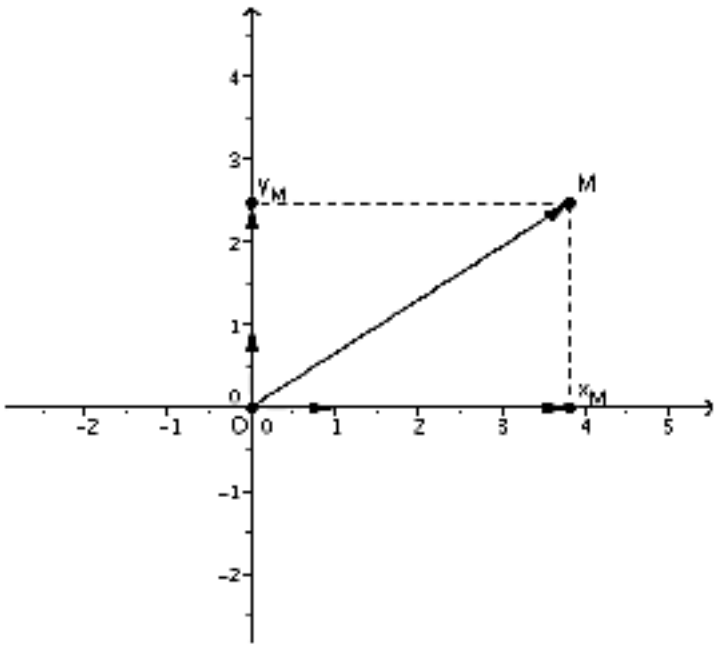
18) REPRÉSENTATION GRAPHIQUE D'UNE FONCTION À UNE VARIABLE.

a) COORDONNÉES DANS LE PLAN.

Dans le plan muni d'un repère, on peut faire correspondre à chaque point M un couple (x_M, y_M) de deux nombres.

Ce couple est appelé couple des coordonnées de M. On écrit : M(x_M, y_M).

x_M est l'abscisse de M et y_M est l'ordonnée de M.



b) REPRÉSENTATION GRAPHIQUE D'UNE FONCTION D'UNE VARIABLE.

Soit f une fonction définie sur une partie D de $\mathbb{R}$.

Dans le plan muni d'un repère, on appelle représentation graphique de f l'ensemble des points dont les coordonnées sont $(x, f(x))$, avec $x \in$ D.

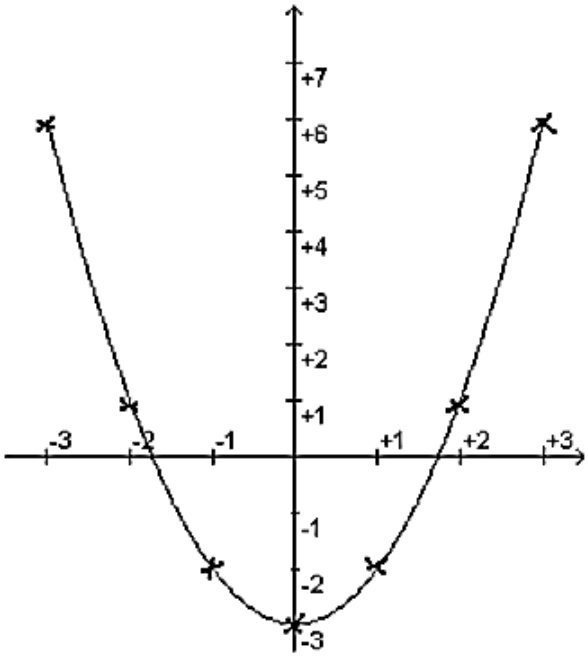
Exemple :

Considérons la fonction f : $x \mapsto x^2 - 3$ définie sur l'intervalle [-3 ;3] et construisons sa représentation graphique.

Pour effectuer cette construction, nous commencerons par calculer un certain nombre d'images. Les résultats sont inscrits dans un tableau de valeurs :

x	-3	-2	-1	0	1	2	3
$f(x)$	6	1	-2	-3	-2	1	6

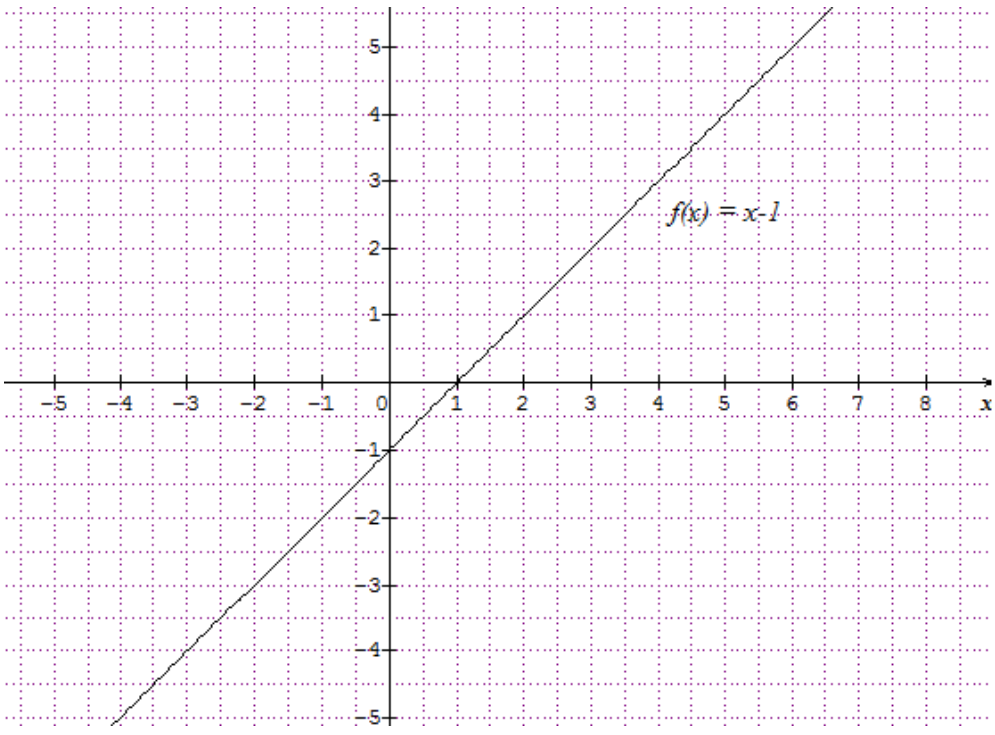
Dans le plan muni de son repère, on place les points de coordonnées $(x, f(x))$, puis on les relie par une courbe.



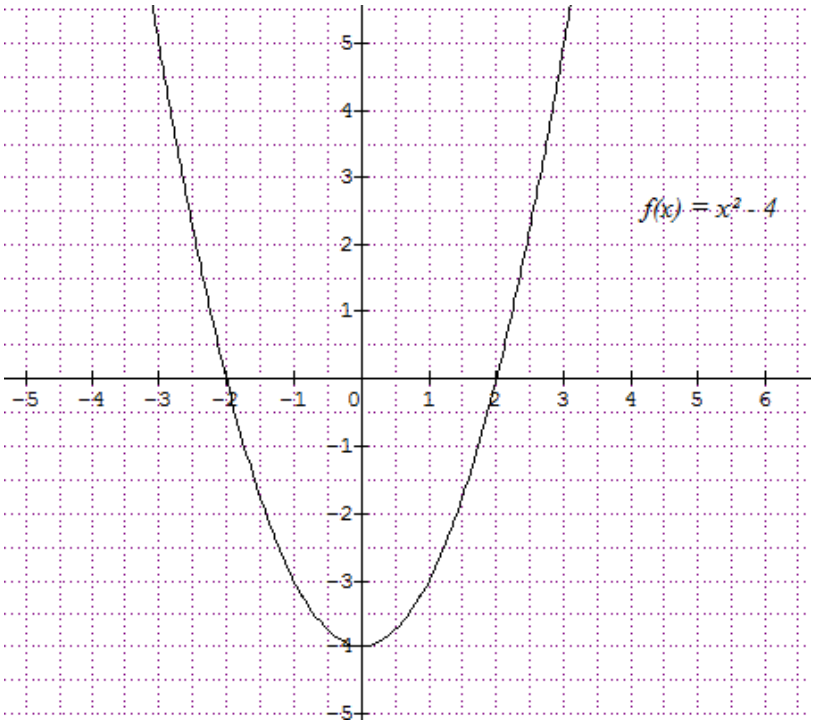
c) UTILISATION DU LOGICIEL SINEQUANON.

À l'aide du logiciel, représentez le graphique des fonctions suivantes :

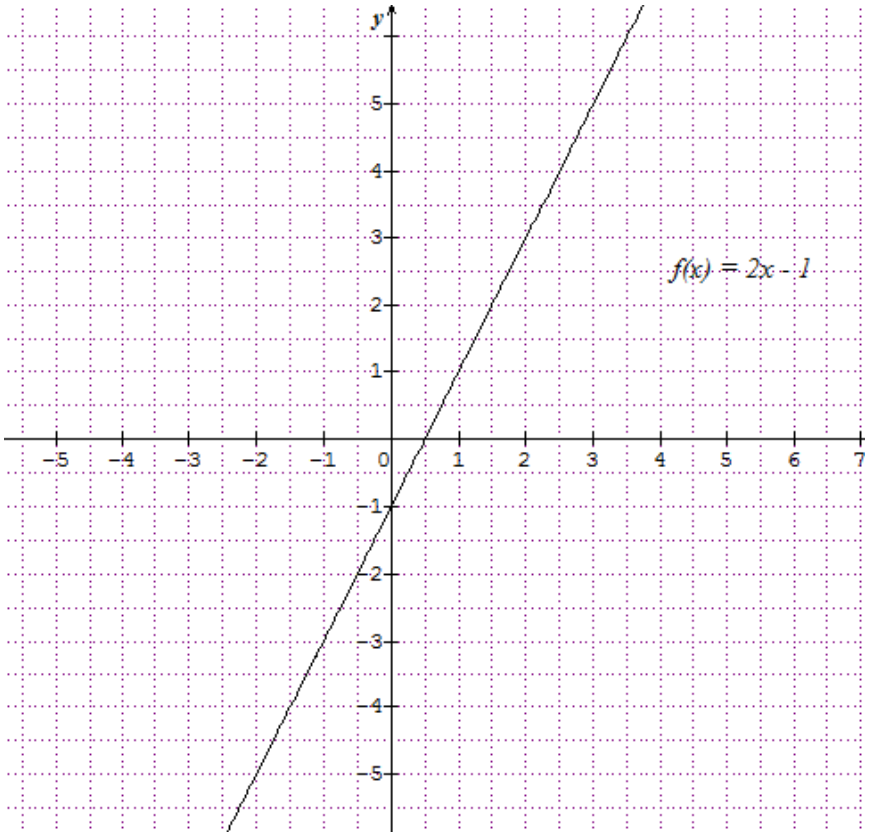
$f : x \mapsto x - 1$



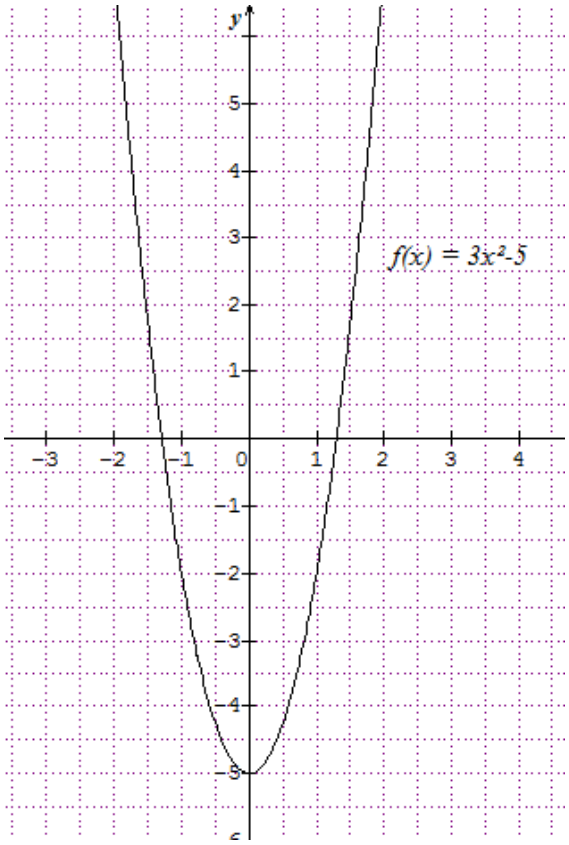
$f : x \mapsto x^2 - 4$



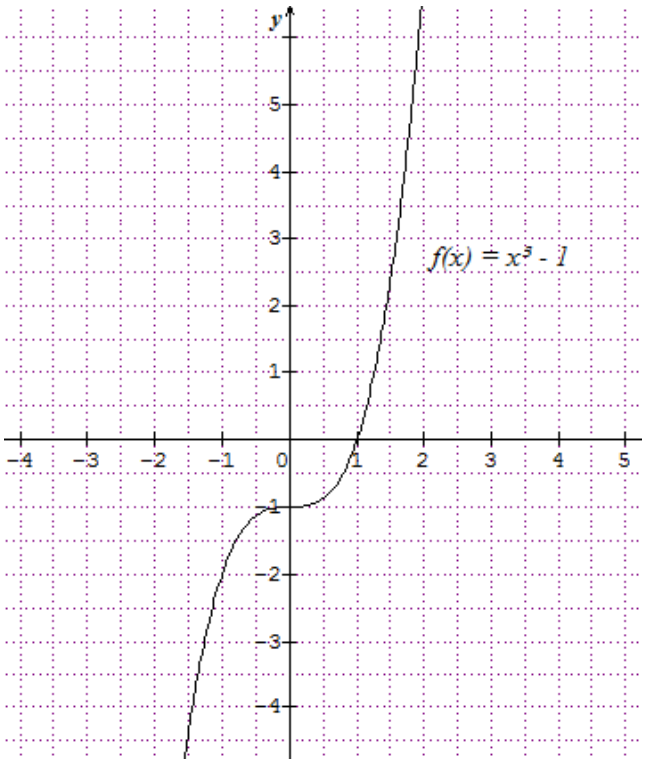
$f : x \mapsto 2x - 1$



$f : x \mapsto 3x^2 - 5$



$f : x \mapsto x^3 - 1$



19) REPRÉSENTATION D'INFORMATIONS CONTENUES DANS UN TEXTE SOUS FORME DE TABLEAU, SCHÉMA OU GRAPHIQUE.

La logique est une forme d'opération de la pensée qui nous permet de raisonner. C'est par exemple la démarche qui vous permettrait de résoudre l'énigme suivante :
Un homme regarde un portrait et dit : « Je n'ai ni frère ni sœur mais le père de cet homme est le fils de mon père ». Qui est représenté sur le portrait ?

Les questions de logique sont parfois moins énigmatiques. :

S'il fait beau ce soir et si j'ai fini ma préparation, j'irai me promener.

Nous tenterons de les ramener à des choix simples où les affirmations sont soit vraies soit fausses, les réponses aux questions sont oui ou non, il n'y a pas de valeurs intermédiaires.
La promenade est liée à deux conditions : la météo et le travail qui reste à faire. Les différentes situations sont représentées dans ce qu'on appelle une table de vérité :

Il fera beau	Mon travail sera achevé	J'irai me promener
Oui	Oui	Oui
Oui	Non	Non
Non	Oui	Non
Non	Non	Non

Essayez avec la proposition suivante :

L'accusé sera disculpé si l'enquête révèle qu'il s'agit d'un suicide ou s'il peut faire la preuve qu'il était ailleurs au moment des faits[55].

Dans le premier cas, la promenade dépend de deux conditions qui doivent être simultanées.
Dans le second cas, une seule condition suffit pour disculper l'accusé.

Exercices :

1. Julie ira au bal ce soir si elle range sa chambre ou nettoie la cuisine.[56]

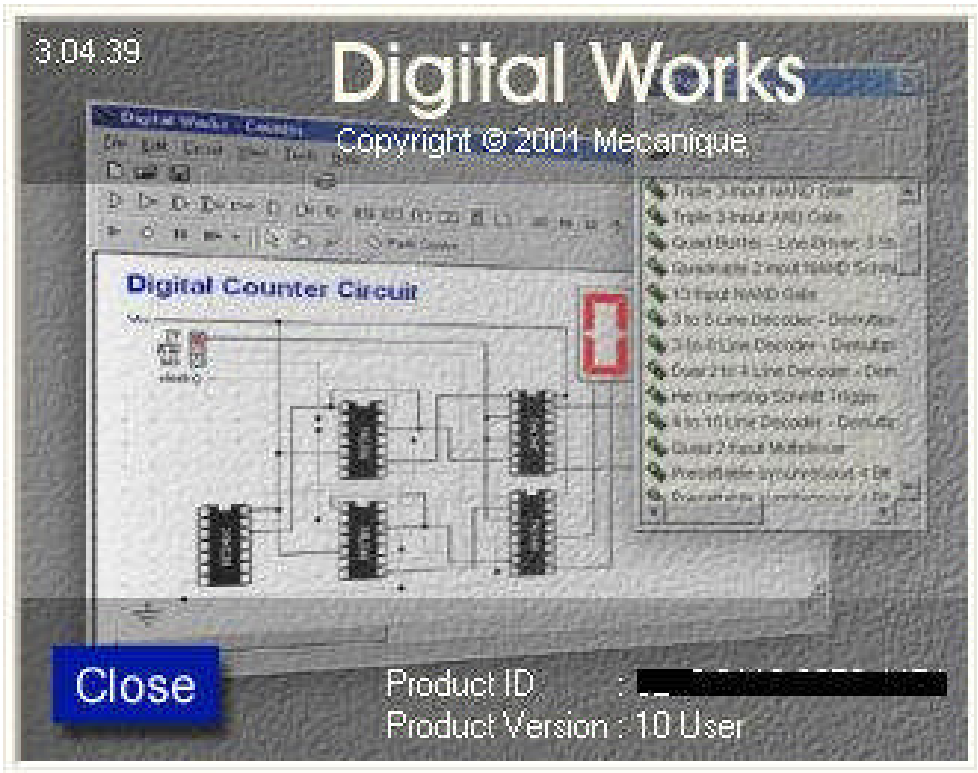
2. J'irai en vacances au mois de juillet si je gagne au Lotto et si mon patron m'accorde mes congés.[57]

3. J'irai au restaurant ce soir si je rentre du travail avant 18h00 et si Jean et/ou Christine sont disponibles.[58]

4. J'irai courir ce soir si je rentre du travail avant 18h00 et s'il ne pleut pas et si je retrouve mes baskets.[59]

20) INTERPRÉTATION DES INFORMATIONS CONTENUES DANS UN GRAPHIQUE EN LANGAGE MATHÉMATIQUE ET EN LANGAGE COURANT.

21) DIGITAL WORKS 3.04.39 : GUIDE PRATIQUE.



a) BRÈVE INTRODUCTION.

Le logiciel « DIGITAL WORKS » permet de schématiser, en respectant les symboles « standard », des circuits logiques et, ensuite, de les tester. Les schémas réalisés peuvent être sauvegardés ou simplement copiés pour être ensuite collés dans un document de présentation (comme MS WORD, par exemple).

Table des matières.

1) Notation scientifique. 4
   a) Introduction. 4
   b) Mode de résolution. 4
   c) Exemples. 5
   d) Entraînement. 5
2) numération et bases de numérations. 6
   a) Système de numération. 6
   b) Les bases de numération. 6
3) Écriture des nombres entiers ou Numération de position. 8
   a) Écriture des nombres. 8
   b) Les chiffres romains. 8
   c) Numération de position. 9
4) Calcul de la valeur d'un nombre quelle que soit sa base. 12
5) Numération binaire. 13
   a) Poids des bits. 13
   b) Un peu de vocabulaire… 14
   c) Zéro à gauche : 14
   d) Petit quiz : 15
6) Pesée d'un nombre en binaire. 16
7) Numération hexadécimale. 18
   a) Chiffres hexadécimaux. 18
   b) Pourquoi utiliser la base 16 ? 18
      - Hexadécimal / Binaire – Mode d'emploi. 19
      - Conversion Hexadécimal ↦ Binaire. 20
      - Conversion binaire ↦ Hexadécimal. 20
   c) Exercices. 20
8) Nombres de codes possibles avec N chiffres en base quelconque. 22
   a) En décimal (= en base 10). 22
   b) En hexadécimal (= en base 16). 22
   c) En binaire (= en base 2). 22
   d) Tailles des nombres entiers et nombre de codes réalisables. 22
   e) Conclusion. 22

9) Puissances de 2. 23
a) Préfixes pour représenter les puissances de 2^{10} ou 10^3. 23
b) Pour les informaticiens, 1 k est-ce 1 000 ou 1 024 ? 24
c) Calculs approximatifs de 2^n avec n > 10. 24
10) Convertir un nombre entier dans une base quelconque. 27
a) Rappel : valeur de chaque chiffre. 27
b) Méthode intuitive : de gauche à droite. 27
c) Méthode systématique : de droite à gauche. 28
d) Conclusions. 29
e) Autre méthode pour convertir une base B en base 10 = « Méthode de Horner ». 29
f) Exercices. 30
11) Nombres signés. 31
a) Comment calculer les codes des nombres négatifs ? 34
b) Analogie en décimal. 35
c) La valeur du bit de signe. 35
d) Extension de la taille d'un nombre signé. 36
e) Exercices. 36
- Quiz – Nombres signés. 36
12) Opérations arithmétiques en binaire. 38
a) Addition. 38
b) Soustraction. 39
c) Multiplication. 40
d) Division. 40
e) Conclusion. 41
13) Logique booléenne. 42
a) Introduction. 42
b) Variables logiques. 42
14) Fonctions logiques de base. 44
a) La fonction ET. 45
- Table de vérité. 45
- Équation logique de la fonction ET. 45
b) La fonction OU. 46
- Table de vérité. 46
- Équation logique de la fonction OU. 46

c) La fonction NON. 47
- Table de vérité. 47
- Équation logique de la fonction NON. 47

15) Combinaisons de fonctions logiques. 48
a) Combinaisons des 3 fonctions de base. 48
- Correspondance entre schémas et équations logiques. 48
- Tables de vérité. 50

b) Exercices. 51
16) Les portes logiques. 57
a) Fonctions logiques de base. 57
- Porte AND. 57
- Porte OR. 58
- Porte NOT. 59

b) Combinaisons des fonctions logiques de base. 60
- Porte NAND (NON ET). 60
- Porte NOR (NON OU) 61
- Porte XOR (OU exclusif). 62
- Porte XNOR (NON OU exclusif). 63
- Porte XOR à plusieurs entrées. 63

c) Exercice d'association 64
17) unités informatiques de mesure de quantité d'informations (bit, octet et multiples), de débit (bps, bauds), de fréquence (Hz et multiples). 65
a) Le système de numération binaire 65
b) Les multiples informatiques 65
c) Les unités de capacité 66
d) Les unités de fréquence 66
e) Les unités de temps 66
f) Les unités de vitesse 67
18) Représentation graphique d'une fonction à une variable. 68
a) Coordonnées dans le plan. 68
b) Représentation graphique d'une fonction d'une variable. 68
c) Utilisation du logiciel SINEQUANON. 69
19) Représentation d'informations contenues dans un texte sous forme de tableau, schéma ou graphique. 72

20) Interprétation des informations contenues dans un graphique en langage mathématique et en langage courant. 74
21) DIGITAL WORKS 3.04.39 : guide pratique. 75
a) Brève introduction 75

1

Entourer les notations scientifiques parmi les écritures suivantes :

$\boxed{2{,}25 \times 10^{4}}$	$0{,}14 \times 10^{3}$	$25{,}3 \times 10^{4}$	$\boxed{2{,}5 \times 10^{-2}}$	$10{,}34 \times 10^{5}$
$3^{5} \times 10^{4}$	$5{,}3 \times 2^{10}$	$\boxed{1{,}01 \times 10^{-4}}$	$4{,}1 \times 2^{3}$	$\boxed{10^{2} \times 4{,}8}$

2

Compléter pour que l'écriture du nombre soit sa notation scientifique :

3 745,6	=	3,7456	x	10^{3}	2 715 000	=	2,715	x	10^{6}
834,75	=	8,3475	x	10^{2}	0,0875	=	8,75	x	10^{-2}
0,000004	=	4	x	10^{-6}					

3

Écrire les nombres en utilisant la notation scientifique :

5 000	=	5×10^{3}	0,25	=	$2{,}5 \times 10^{-1}$
1 000 000	=	1×10^{6}	0,00025	=	$2{,}5 \times 10^{-4}$
235,7	=	$2{,}357 \times 10^{2}$	1010	=	$1{,}01 \times 10^{3}$
125 000	=	$1{,}25 \times 10^{5}$			

4

Les composants qui sont à la base des fonctions logiques agissent comme des contacts : Ouverts ou Fermés.
Ils ne peuvent représenter que 2 états, symbolisés par 0 et 1.

5

Il y a 8 chiffres en octal. Ils vont de 0 à 7.

6

Le poids d'un chiffre est effectivement la base exposant la position. Ce qui donne ici 10^{2} et cela vaut 100.

7

Le chiffre de position 1 est celui des dizaines dont le chiffre 6 dans l'exemple.

8

La base exposant la position = $8^2 = 64$.

9

$1000_{(2)} = 1 \times 2^3 = 8$

10

$0101_{(2)} = 1 \times 2^2 + 1 \times 2^0 = 5$

11

$0100_{(2)} = 1 \times 2^2 = 4$

12

$0001\ 0000_{(2)} = 1 \times 2^4 = 16$

13

Vrai
Le chiffre le plus à droite est le chiffre en position 0.
Son poids est la base exposant 0.
Cela donne bien un poids de 1.

14

$1010_{(2)} = 1 \times 2^3 + 1 \times 2^1 = 8 + 2 = 10$

15

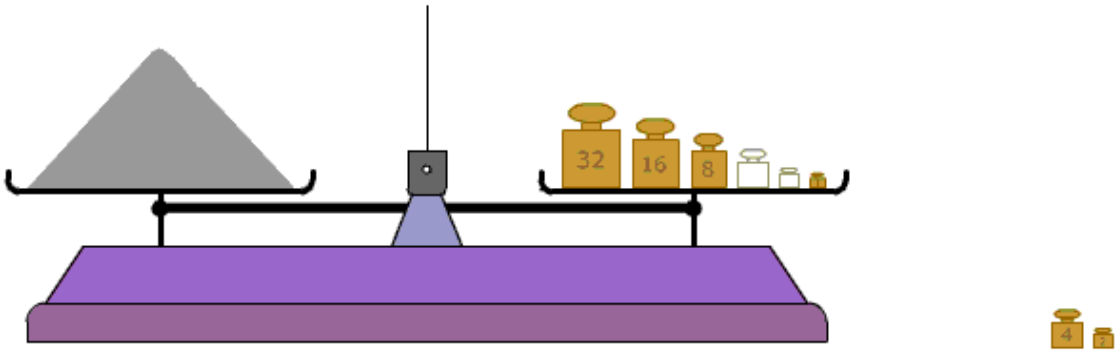
$0010_{(2)} = 1 \times 2^1 = 2$

16

$0111_{(2)} = 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 4 + 2 + 1 = 7$

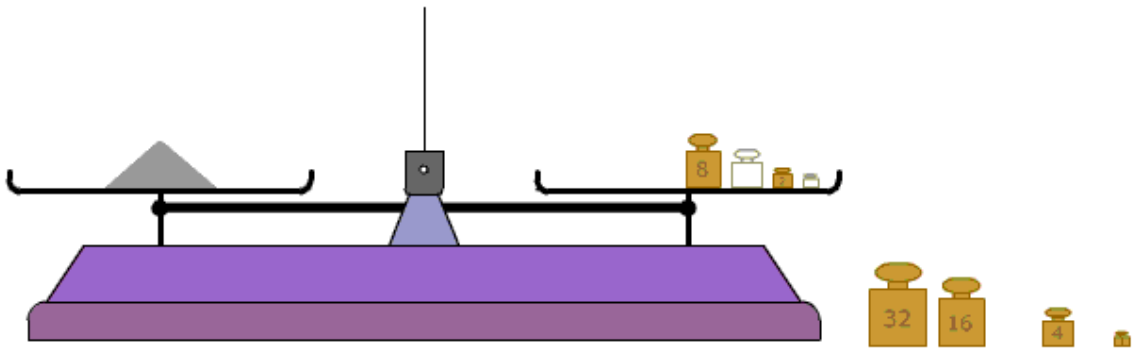
17

$57_{(10)} = 111001_{(2)}$



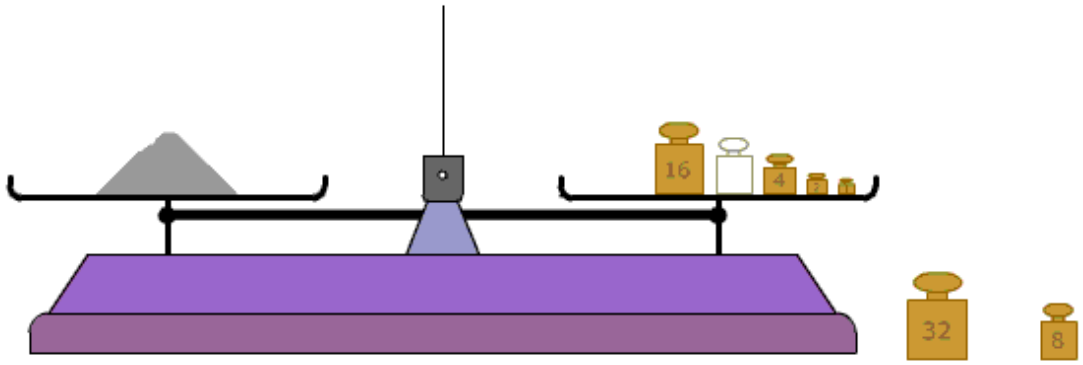
18

$10_{(10)} = 1010_{(2)}$



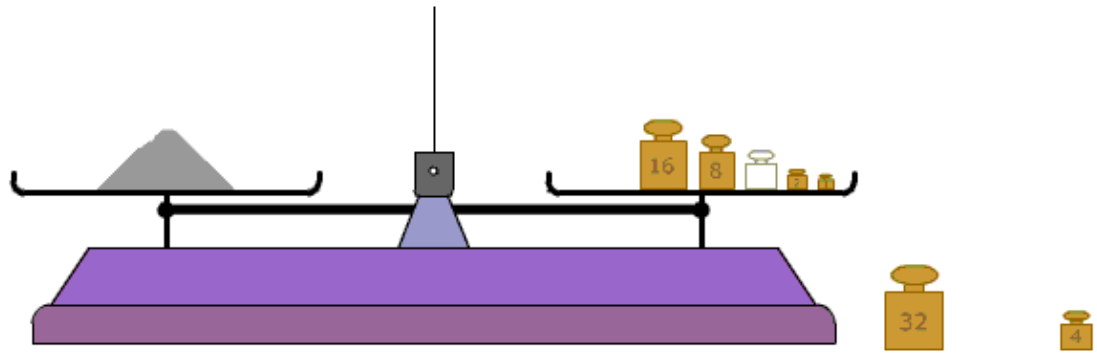
19

$23_{(10)} = 10111_{(2)}$



20

$27_{(10)} = 11011_{(2)}$



21

Réponse C : un chiffre binaire
« Bit » vient de de la contraction de Binary digit
Ce qui signifie "CHIFFRE BINAIRE"

22

Le processeur travaille en binaire car il est construit à partir de composants qui ne peuvent prendre que 2 états distincts que l'on représente par 0 et 1

23

$2^4 = 16$ → 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111

24

$2^8 = 256$ mais ils vont de 0 à 255 donc la plus grande valeur est 255.

25

$2^{10} = 1\ 024$

26

Réponse B.
Il est en effet plus facile de retenir A674 que 1010 0110 0111 0100
Ces deux codes sont équivalents mais le premier est donné en hexadécimal, le second en binaire est une suite interminable de 0 et de 1.

27

Réponse C : 10

28

F

29

Un octet ou un byte

30

160
$A0_{(16)} = 10 \times 16 + 0 = 160$

31

Il y a 1 000 µs dans une ms.

32

1 Angström = 10^{-10} mètre = 0,1 nanomètre = 100 picomètres
Le nanomètre est davantage utilisé que l'angström. Un nanomètre équivaut à un milliardième de mètre, soit 10^{-9} mètre, soit 10 angströms.

33

$2^{12} = 2^2 \times 2^{10} = 4 \times 1\ 024 = 4\ 096 \approx 4\ 000$
$2^{32} = 2^2 \times 2^{10} \times 2^{10} \times 2^{10} = 4 \times 1\ 024 \times 1\ 024 \times 1\ 024 \approx 2\ 000\ 000\ 000$
$2^{16} = 2^6 \times 2^{10} = 64 \times 1\ 024 \approx 64\ 000$
$2^{27} = 2^7 \times 2^{10} \times 2^{10} = 128 \times 1\ 024 \times 1\ 024 \approx 128\ 000\ 000$
$2^{36} = 2^6 \times 2^{30} \approx 2^6 \times 10^9 = 64 \times 1\ 000\ 000\ 000 = 64\ 000\ 000\ 000$

34

$26^3 \times 10^3 = 26 \times 26 \times 26 \times 1000 = 17\,576\,000$ possibilités.

35

Réponse C : 1M

$2^{20} = 2^{10} \times 2^{10} = 1K \times 1K = 1M$

1 Kilo x 1 Kilo = 1 Méga

36

Réponse D : 16 M4

$2^{24} = 2^4 \times 2^{10} \times 2^{10} = 16 \times 1K \times 1K = 16$ m (Méga)

37

Réponse D : 4 milliards

$2^{32} = 2^2 \times 2^{10} \times 2^{10} \times 2^{10} = 4 \times 1K \times 1K \times 1K =$G (4 milliards)

38

$2^{12} = 2^2 \times 2^{10}$

$2^2 = 4$ et $2^{10} = 1$ kilo

On a donc $2^{12} = 4$ K

39

Réponse C : 100 000 000 bits/s

1 Méga = 1 Million = 10^6

40

1 024

41

Réponse B : 64 K

$2^{16} = 2^6 \times 2^{10} = 64 \times 1$ K = 64 Kilo

42

Réponse B : 10^3 =1 000 et 2^{10} = 1 024

43

$203_{(8)} = (((2 \times 8) + 0) \times 8) + 3 = 131_{(10)}$

$101010_{(2)} = (((((((1 \times 2) + 0) \times 2) + 1) \times 2) + 0) \times 2) + 1) \times 2) + 0 = 42_{(10)}$

$20A_{(16)} = (((2 \times 16) + 0 \times 16) + 10 = 522_{(10)}$

44

$166_{(10)} = A6_{(16)}$

$100_{(10)} = 1100100_{(2)}$

$100_{(10)} = 64_{(16)}$
$1023_{(10)} = 3FF_{(16)}$
$1023_{(10)} = 0011\ 1111\ 1111_{(2)}$
$C0CA_{(16)} = 49\ 354_{(10)}$
$1011011_{(2)} = 91_{(10)}$
$236_{(8)} = 158_{(10)}$
$FFF_{(16)} = 4\ 095_{(10)}$

45

Réponse C : Bit 7
Le bit de signe est le msb (*Most Significant Bit*)
C'est le bit le plus à gauche.
Puisque les bits sont numérotés de droite à gauche en commençant par le n° 0, le huitième bit est donc bien le bit 7

46

Dans un octet, le bit le plus à gauche sert de signe.
Il ne reste donc que 7 bits pour représenter tous les nombres positifs
$2^7 = 128$ combinaisons possibles
La combinaison la plus petite 0000 0000 à pour valeur 0
Le nombre positif le plus grand s'écrit 0111 1111 en binaire, ce qui vaut +127 en base 10

47

-1

48

255

49

$80_{(16)}$ est la valeur la plus négative qu'on puisse écrire avec un nombre signé de 8 bits
Elle s'écrit 1000 0000 en binaire
$80_{(16)}$ = -128 en base 10

50

FF en hexadécimal signé vaut -1
Pour preuve, dans l'absolu, sans tenir compte des tailles des nombre FF + 1= 100 mais comme le calcul est fait ici sur 1 octet, seuls les deux derniers 00 du résultat sont conservés.

51

FFFF est le code qui précède 0000 lorsqu'on compte avec des nombres de 2 bytes.

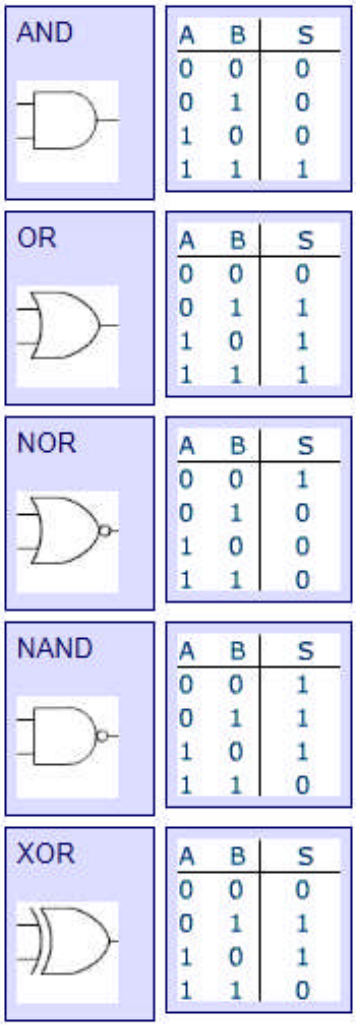
52

7F non signé = 127
7F signé = +127
Ces 2 valeurs sont bien les mêmes.

53

Deux octets = 16 bits. Il s'agit ici d'un nombre signé.
Un bit est donc réservé au signe. Il ne reste donc que 15 bits pour faire tous les codes positifs
Ce qui fait 2^{15} codes différents allant de 0 à 2^{15}-1 = 32 768 – 1 = 32 767

54



55

Il s'agit d'un suicide	Ailleurs au moment des faits	Disculpé
Oui	Oui	Oui
Oui	Non	Oui
Non	Oui	Oui
Non	Non	Non

56

Elle range sa chambre	Elle nettoie la cuisine	Elle ira au bal
Oui	Oui	Oui
Oui	Non	Oui
Non	Oui	Oui
Non	Non	Non

57

Je gagne au lotto	Je suis en congé	Je pars en vacances
Oui	Oui	Oui
Oui	Non	Non
Non	Oui	Non
Non	Non	Non

58

Jean est disponible	Christine est disponible	Je rentre avant 18h00	Je vais au restaurant
Oui	Oui	Oui	Oui
Oui	Oui	Non	Non
Oui	Non	Oui	Oui
Oui	Non	Non	Non
Non	Oui	Oui	Oui
Non	Oui	Non	Non
Non	Non	Oui	Non
Non	Non	Non	Non

59

Je rentre avant 18h00	Il ne pleut pas	Je trouve mes baskets	Je vais courir
Oui	Oui	Oui	Oui
Oui	Oui	Non	Non
Oui	Non	Oui	Non
Oui	Non	Non	Non
Non	Oui	Oui	Non
Non	Oui	Non	Non
Non	Non	Oui	Non
Non	Non	Non	Non